**[MS-RDPEGDI]:**

**Remote Desktop Protocol: Graphics Device Interface (GDI) Acceleration Extensions**

**Intellectual Property Rights Notice for Open Specifications Documentation**

* **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
* **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 make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
* **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
* **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Community Promise](http://go.microsoft.com/fwlink/?LinkId=214448). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
* **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
* **Fictitious Names**. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events 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 specifically described above, whether by implication, estoppel, or otherwise.

**Tools**. The Open Specifications do 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 are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 2/22/2007 | 0.01 |  | Version 0.01 release |
| 6/1/2007 | 1.0 | Major | Updated and revised the technical content. |
| 7/3/2007 | 1.1 | Minor | Minor technical content updates. |
| 7/20/2007 | 1.1.1 | Editorial | Changed language and formatting in the technical content. |
| 8/10/2007 | 1.2 | Minor | Updated content based on feedback. |
| 9/28/2007 | 1.3 | Minor | Made technical and editorial changes based on feedback. |
| 10/23/2007 | 1.3.1 | Editorial | Changed language and formatting in the technical content. |
| 11/30/2007 | 1.4 | Minor | Made technical and editorial changes based on feedback. |
| 1/25/2008 | 2.0 | Major | Updated and revised the technical content. |
| 3/14/2008 | 3.0 | Major | Updated and revised the technical content. |
| 5/16/2008 | 3.0.1 | Editorial | Changed language and formatting in the technical content. |
| 6/20/2008 | 3.1 | Minor | Clarified the meaning of the technical content. |
| 7/25/2008 | 3.2 | Minor | Clarified the meaning of the technical content. |
| 8/29/2008 | 3.3 | Minor | Clarified the meaning of the technical content. |
| 10/24/2008 | 3.3.1 | Editorial | Changed language and formatting in the technical content. |
| 12/5/2008 | 4.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 5.0 | Major | Updated and revised the technical content. |
| 2/27/2009 | 5.0.1 | Editorial | Changed language and formatting in the technical content. |
| 4/10/2009 | 5.0.2 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 6.0 | Major | Updated and revised the technical content. |
| 7/2/2009 | 7.0 | Major | Updated and revised the technical content. |
| 8/14/2009 | 8.0 | Major | Updated and revised the technical content. |
| 9/25/2009 | 8.1 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 9.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 10.0 | Major | Updated and revised the technical content. |
| 1/29/2010 | 11.0 | Major | Updated and revised the technical content. |
| 3/12/2010 | 12.0 | Major | Updated and revised the technical content. |
| 4/23/2010 | 12.0.1 | Editorial | Changed language and formatting in the technical content. |
| 6/4/2010 | 13.0 | Major | Updated and revised the technical content. |
| 7/16/2010 | 14.0 | Major | Updated and revised the technical content. |
| 8/27/2010 | 15.0 | Major | Updated and revised the technical content. |
| 10/8/2010 | 16.0 | Major | Updated and revised the technical content. |
| 11/19/2010 | 17.0 | Major | Updated and revised the technical content. |
| 1/7/2011 | 18.0 | Major | Updated and revised the technical content. |
| 2/11/2011 | 19.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 20.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 20.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/17/2011 | 21.0 | Major | Updated and revised the technical content. |
| 9/23/2011 | 22.0 | Major | Updated and revised the technical content. |
| 12/16/2011 | 23.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 24.0 | Major | Updated and revised the technical content. |
| 7/12/2012 | 25.0 | Major | Updated and revised the technical content. |
| 10/25/2012 | 26.0 | Major | Updated and revised the technical content. |
| 1/31/2013 | 26.1 | Minor | Clarified the meaning of the technical content. |
| 8/8/2013 | 27.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 27.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 27.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 27.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 28.0 | Major | Significantly changed the technical content. |

Table of Contents

[1 Introduction 11](#_Toc423365935)

[1.1 Glossary 11](#_Toc423365936)

[1.2 References 12](#_Toc423365937)

[1.2.1 Normative References 12](#_Toc423365938)

[1.2.2 Informative References 13](#_Toc423365939)

[1.3 Overview 13](#_Toc423365940)

[1.3.1 Accelerated Graphics 13](#_Toc423365941)

[1.3.1.1 Caches 14](#_Toc423365942)

[1.3.1.2 Drawing Orders 14](#_Toc423365943)

[1.3.1.2.1 Primary Drawing Orders 15](#_Toc423365944)

[1.3.1.2.2 Secondary Drawing Orders 15](#_Toc423365945)

[1.3.1.2.3 Alternate Secondary Drawing Orders 16](#_Toc423365946)

[1.3.1.3 Error Conditions 16](#_Toc423365947)

[1.3.2 Bulk Data Compression 16](#_Toc423365948)

[1.3.2.1 RDP 6.0 16](#_Toc423365949)

[1.3.2.2 RDP 6.1 17](#_Toc423365950)

[1.4 Relationship to Other Protocols 17](#_Toc423365951)

[1.5 Prerequisites/Preconditions 17](#_Toc423365952)

[1.6 Applicability Statement 17](#_Toc423365953)

[1.7 Versioning and Capability Negotiation 17](#_Toc423365954)

[1.8 Vendor-Extensible Fields 17](#_Toc423365955)

[1.9 Standards Assignments 17](#_Toc423365956)

[2 Messages 18](#_Toc423365957)

[2.1 Transport 18](#_Toc423365958)

[2.2 Message Syntax 18](#_Toc423365959)

[2.2.1 Capability Sets 18](#_Toc423365960)

[2.2.1.1 Color Table Cache Capability Set (TS\_COLORTABLE\_CAPABILITYSET) 18](#_Toc423365961)

[2.2.1.2 DrawNineGrid Cache Capability Set (TS\_DRAW\_NINEGRID\_CAPABILITYSET) 18](#_Toc423365962)

[2.2.1.3 Draw GDI+ Capability Set (TS\_DRAW\_GDIPLUS\_CAPABILITYSET) 19](#_Toc423365963)

[2.2.1.3.1 GDI+ Cache Entries (TS\_GDIPLUS\_CACHE\_ENTRIES) 21](#_Toc423365964)

[2.2.1.3.2 GDI+ Cache Chunk Size (TS\_GDIPLUS\_CACHE\_CHUNK\_SIZE) 21](#_Toc423365965)

[2.2.1.3.3 GDI+ Image Cache Properties (TS\_GDIPLUS\_IMAGE\_CACHE\_PROPERTIES) 22](#_Toc423365966)

[2.2.2 Accelerated Graphics 22](#_Toc423365967)

[2.2.2.1 Orders Update (TS\_UPDATE\_ORDERS\_PDU\_DATA) 22](#_Toc423365968)

[2.2.2.1.1 Drawing Order (DRAWING\_ORDER) 23](#_Toc423365969)

[2.2.2.2 Fast-Path Orders Update (TS\_FP\_UPDATE\_ORDERS) 23](#_Toc423365970)

[2.2.2.2.1 Drawing Order (DRAWING\_ORDER) 24](#_Toc423365971)

[2.2.2.2.1.1 Primary Drawing Orders 24](#_Toc423365972)

[2.2.2.2.1.1.1 Common Data Types 24](#_Toc423365973)

[2.2.2.2.1.1.1.1 Coord Field (COORD\_FIELD) 24](#_Toc423365974)

[2.2.2.2.1.1.1.2 One-Byte Header Variable Field (VARIABLE1\_FIELD) 25](#_Toc423365975)

[2.2.2.2.1.1.1.3 Two-Byte Header Variable Field (VARIABLE2\_FIELD) 25](#_Toc423365976)

[2.2.2.2.1.1.1.4 Delta-Encoded Points (DELTA\_PTS\_FIELD) 25](#_Toc423365977)

[2.2.2.2.1.1.1.5 Delta-Encoded Rectangles (DELTA\_RECTS\_FIELD) 26](#_Toc423365978)

[2.2.2.2.1.1.1.6 Binary Raster Operation (ROP2\_OPERATION) 27](#_Toc423365979)

[2.2.2.2.1.1.1.7 Ternary Raster Operation Index (ROP3\_OPERATION\_INDEX) 28](#_Toc423365980)

[2.2.2.2.1.1.1.8 Generic Color (TS\_COLOR) 42](#_Toc423365981)

[2.2.2.2.1.1.1.9 Fill Mode (FILL\_MODE) 43](#_Toc423365982)

[2.2.2.2.1.1.2 Primary Drawing Order (PRIMARY\_DRAWING\_ORDER) 43](#_Toc423365983)

[2.2.2.2.1.1.2.1 DstBlt (DSTBLT\_ORDER) 47](#_Toc423365984)

[2.2.2.2.1.1.2.2 MultiDstBlt (MULTI\_DSTBLT\_ORDER) 48](#_Toc423365985)

[2.2.2.2.1.1.2.3 PatBlt (PATBLT\_ORDER) 49](#_Toc423365986)

[2.2.2.2.1.1.2.4 MultiPatBlt (MULTI\_PATBLT\_ORDER) 52](#_Toc423365987)

[2.2.2.2.1.1.2.5 OpaqueRect (OPAQUERECT\_ORDER) 54](#_Toc423365988)

[2.2.2.2.1.1.2.6 MultiOpaqueRect (MULTI\_OPAQUERECT\_ORDER) 55](#_Toc423365989)

[2.2.2.2.1.1.2.7 ScrBlt (SCRBLT\_ORDER) 55](#_Toc423365990)

[2.2.2.2.1.1.2.8 MultiScrBlt (MULTI\_SCRBLT\_ORDER) 56](#_Toc423365991)

[2.2.2.2.1.1.2.9 MemBlt (MEMBLT\_ORDER) 57](#_Toc423365992)

[2.2.2.2.1.1.2.10 Mem3Blt (MEM3BLT\_ORDER) 59](#_Toc423365993)

[2.2.2.2.1.1.2.11 LineTo (LINETO\_ORDER) 61](#_Toc423365994)

[2.2.2.2.1.1.2.12 SaveBitmap (SAVEBITMAP\_ORDER) 62](#_Toc423365995)

[2.2.2.2.1.1.2.13 GlyphIndex (GLYPHINDEX\_ORDER) 63](#_Toc423365996)

[2.2.2.2.1.1.2.14 FastIndex (FASTINDEX\_ORDER) 67](#_Toc423365997)

[2.2.2.2.1.1.2.15 FastGlyph (FASTGLYPH\_ORDER) 68](#_Toc423365998)

[2.2.2.2.1.1.2.16 PolygonSC (POLYGON\_SC\_ORDER) 70](#_Toc423365999)

[2.2.2.2.1.1.2.17 PolygonCB (POLYGON\_CB\_ORDER) 71](#_Toc423366000)

[2.2.2.2.1.1.2.18 Polyline (POLYLINE\_ORDER) 72](#_Toc423366001)

[2.2.2.2.1.1.2.19 EllipseSC (ELLIPSE\_SC\_ORDER) 73](#_Toc423366002)

[2.2.2.2.1.1.2.20 EllipseCB (ELLIPSE\_CB\_ORDER) 74](#_Toc423366003)

[2.2.2.2.1.1.2.21 DrawNineGrid (DRAWNINEGRID\_ORDER) 75](#_Toc423366004)

[2.2.2.2.1.1.2.22 MultiDrawNineGrid (MULTI\_DRAWNINEGRID\_ORDER) 76](#_Toc423366005)

[2.2.2.2.1.2 Secondary Drawing Orders 77](#_Toc423366006)

[2.2.2.2.1.2.1 Common Data Types 77](#_Toc423366007)

[2.2.2.2.1.2.1.1 Secondary Drawing Order Header (SECONDARY\_DRAWING\_ORDER\_HEADER) 77](#_Toc423366008)

[2.2.2.2.1.2.1.2 Two-Byte Unsigned Encoding (TWO\_BYTE\_UNSIGNED\_ENCODING) 78](#_Toc423366009)

[2.2.2.2.1.2.1.3 Two-Byte Signed Encoding (TWO\_BYTE\_SIGNED\_ENCODING) 78](#_Toc423366010)

[2.2.2.2.1.2.1.4 Four-Byte Unsigned Encoding (FOUR\_BYTE\_UNSIGNED\_ENCODING) 79](#_Toc423366011)

[2.2.2.2.1.2.2 Cache Bitmap - Revision 1 (CACHE\_BITMAP\_ORDER) 80](#_Toc423366012)

[2.2.2.2.1.2.3 Cache Bitmap - Revision 2 (CACHE\_BITMAP\_REV2\_ORDER) 81](#_Toc423366013)

[2.2.2.2.1.2.4 Cache Color Table (CACHE\_COLOR\_TABLE\_ORDER) 84](#_Toc423366014)

[2.2.2.2.1.2.4.1 Color Quad (TS\_COLOR\_QUAD) 84](#_Toc423366015)

[2.2.2.2.1.2.5 Cache Glyph - Revision 1 (CACHE\_GLYPH\_ORDER) 85](#_Toc423366016)

[2.2.2.2.1.2.5.1 Cache Glyph Data (TS\_CACHE\_GLYPH\_DATA) 86](#_Toc423366017)

[2.2.2.2.1.2.6 Cache Glyph - Revision 2 (CACHE\_GLYPH\_REV2\_ORDER) 86](#_Toc423366018)

[2.2.2.2.1.2.6.1 Cache Glyph Data - Revision 2 (TS\_CACHE\_GLYPH\_DATA\_REV2) 87](#_Toc423366019)

[2.2.2.2.1.2.7 Cache Brush (CACHE\_BRUSH\_ORDER) 88](#_Toc423366020)

[2.2.2.2.1.2.7.1 Compressed Color Brush (COMPRESSED\_COLOR\_BRUSH) 90](#_Toc423366021)

[2.2.2.2.1.2.8 Cache Bitmap - Revision 3 (CACHE\_BITMAP\_REV3\_ORDER) 94](#_Toc423366022)

[2.2.2.2.1.3 Alternate Secondary Drawing Orders 96](#_Toc423366023)

[2.2.2.2.1.3.1 Common Data Types 96](#_Toc423366024)

[2.2.2.2.1.3.1.1 Alternate Secondary Drawing Order Header (ALTSEC\_DRAWING\_ORDER\_HEADER) 96](#_Toc423366025)

[2.2.2.2.1.3.2 Create Offscreen Bitmap (CREATE\_OFFSCR\_BITMAP\_ORDER) 97](#_Toc423366026)

[2.2.2.2.1.3.2.1 Offscreen Cache Delete List (OFFSCR\_DELETE\_LIST) 98](#_Toc423366027)

[2.2.2.2.1.3.3 Switch Surface (SWITCH\_SURFACE\_ORDER) 98](#_Toc423366028)

[2.2.2.2.1.3.4 Create NineGrid Bitmap (CREATE\_NINEGRID\_BITMAP\_ORDER) 99](#_Toc423366029)

[2.2.2.2.1.3.4.1 NineGrid Bitmap Information (NINEGRID\_BITMAP\_INFO) 100](#_Toc423366030)

[2.2.2.2.1.3.4.1.1 Color Reference (TS\_COLORREF) 101](#_Toc423366031)

[2.2.2.2.1.3.5 Stream Bitmap Orders 101](#_Toc423366032)

[2.2.2.2.1.3.5.1 Stream Bitmap First (STREAM\_BITMAP\_FIRST\_ORDER) 101](#_Toc423366033)

[2.2.2.2.1.3.5.2 Stream Bitmap Next (STREAM\_BITMAP\_NEXT\_ORDER) 102](#_Toc423366034)

[2.2.2.2.1.3.6 GDI+ Orders 103](#_Toc423366035)

[2.2.2.2.1.3.6.1 Common Data Types 103](#_Toc423366036)

[2.2.2.2.1.3.6.1.1 GDI+ Cache Type (DRAW\_GDIPLUS\_CACHE\_TYPE) 103](#_Toc423366037)

[2.2.2.2.1.3.6.2 Draw GDI+ Cache First (DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER) 104](#_Toc423366038)

[2.2.2.2.1.3.6.3 Draw GDI+ Cache Next (DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER) 105](#_Toc423366039)

[2.2.2.2.1.3.6.4 Draw GDI+ Cache End (DRAW\_GDIPLUS\_CACHE\_END\_ORDER) 106](#_Toc423366040)

[2.2.2.2.1.3.6.5 Draw GDI+ First (DRAW\_GDIPLUS\_FIRST\_ORDER) 106](#_Toc423366041)

[2.2.2.2.1.3.6.6 Draw GDI+ Next (DRAW\_GDIPLUS\_NEXT\_ORDER) 107](#_Toc423366042)

[2.2.2.2.1.3.6.7 Draw GDI+ End (DRAW\_GDIPLUS\_END\_ORDER) 108](#_Toc423366043)

[2.2.2.2.1.3.7 Frame Marker 109](#_Toc423366044)

[2.2.2.3 Error Conditions 109](#_Toc423366045)

[2.2.2.3.1 Client Bitmap Cache Error PDU 109](#_Toc423366046)

[2.2.2.3.1.1 Bitmap Cache Error PDU Data (TS\_BITMAP\_CACHE\_ERROR\_PDU) 110](#_Toc423366047)

[2.2.2.3.1.1.1 Bitmap Cache Error Info (TS\_BITMAP\_CACHE\_ERROR\_INFO) 111](#_Toc423366048)

[2.2.2.3.2 Client Offscreen Bitmap Cache Error PDU 111](#_Toc423366049)

[2.2.2.3.2.1 Offscreen Bitmap Cache Error PDU Data (TS\_OFFSCRCACHE\_ERROR\_PDU) 112](#_Toc423366050)

[2.2.2.3.3 Client DrawNineGrid Cache Error PDU 113](#_Toc423366051)

[2.2.2.3.3.1 DrawNineGrid Cache Error PDU Data (TS\_DRAWNINEGRID\_ERROR\_PDU) 114](#_Toc423366052)

[2.2.2.3.4 Client GDI+ Error PDU 115](#_Toc423366053)

[2.2.2.3.4.1 GDI+ Error PDU Data (TS\_DRAWGDIPLUS\_ERROR\_PDU) 116](#_Toc423366054)

[2.2.2.4 Bulk Data Compression 116](#_Toc423366055)

[2.2.2.4.1 RDP 6.1 Compressed Data (RDP61\_COMPRESSED\_DATA) 116](#_Toc423366056)

[2.2.2.4.1.1 RDP 6.1 Match Details (RDP61\_MATCH\_DETAILS) 118](#_Toc423366057)

[2.2.2.5 Bitmap Compression 118](#_Toc423366058)

[2.2.2.5.1 RDP 6.0 Bitmap Compressed Bitmap Stream (RDP6\_BITMAP\_STREAM) 118](#_Toc423366059)

[2.2.2.5.1.1 RDP 6.0 RLE Segments (RDP6\_RLE\_SEGMENTS) 120](#_Toc423366060)

[2.2.2.5.1.2 RDP 6.0 RLE Segment (RDP6\_RLE\_SEGMENT) 120](#_Toc423366061)

[3 Protocol Details 122](#_Toc423366062)

[3.1 Common Details 122](#_Toc423366063)

[3.1.1 Abstract Data Model 122](#_Toc423366064)

[3.1.1.1 Caches 122](#_Toc423366065)

[3.1.1.1.1 Bitmap Caches 122](#_Toc423366066)

[3.1.1.1.2 Glyph and Fragment Caches 123](#_Toc423366067)

[3.1.1.1.3 Color Table Cache 124](#_Toc423366068)

[3.1.1.1.4 Brush Caches 124](#_Toc423366069)

[3.1.1.1.5 Offscreen Bitmap Cache 124](#_Toc423366070)

[3.1.1.1.6 NineGrid Bitmap Cache 125](#_Toc423366071)

[3.1.1.1.7 GDI+ Caches 125](#_Toc423366072)

[3.1.2 Timers 125](#_Toc423366073)

[3.1.3 Initialization 125](#_Toc423366074)

[3.1.4 Higher-Layer Triggered Events 125](#_Toc423366075)

[3.1.5 Processing Events and Sequencing Rules 125](#_Toc423366076)

[3.1.6 Timer Events 125](#_Toc423366077)

[3.1.7 Other Local Events 126](#_Toc423366078)

[3.1.8 Bulk Data Compression 126](#_Toc423366079)

[3.1.8.1 RDP 6.0 126](#_Toc423366080)

[3.1.8.1.1 Abstract Data Model 126](#_Toc423366081)

[3.1.8.1.2 Compressing Data 126](#_Toc423366082)

[3.1.8.1.3 Decompressing Data 126](#_Toc423366083)

[3.1.8.1.4 Wire Format 127](#_Toc423366084)

[3.1.8.1.4.1 Literal, EOS, and Copy-Offset Tables 127](#_Toc423366085)

[3.1.8.1.4.2 Length-of-Match Tables 131](#_Toc423366086)

[3.1.8.1.4.3 Encoding the Logically Compressed Stream 133](#_Toc423366087)

[3.1.8.1.4.3.1 Encoding the Copy-Offset 134](#_Toc423366088)

[3.1.8.1.4.3.1.1 Examples of Copy-Offset Encoding 135](#_Toc423366089)

[3.1.8.1.4.3.2 Encoding the Length-of-Match 135](#_Toc423366090)

[3.1.8.1.4.3.2.1 Examples of Length-of-Match Encoding 135](#_Toc423366091)

[3.1.8.1.4.4 Decoding a Compressed Stream 136](#_Toc423366092)

[3.1.8.1.4.4.1 Decoding the Copy-Offset 137](#_Toc423366093)

[3.1.8.1.4.4.1.1 Examples of Copy-Offset Decoding 137](#_Toc423366094)

[3.1.8.1.4.4.2 Decoding the Length-of-Match 137](#_Toc423366095)

[3.1.8.1.4.4.2.1 Examples of Length-of-Match Decoding 138](#_Toc423366096)

[3.1.8.2 RDP 6.1 138](#_Toc423366097)

[3.1.8.2.1 Abstract Data Model 138](#_Toc423366098)

[3.1.8.2.2 Compressing Data 141](#_Toc423366099)

[3.1.8.2.2.1 Data Compression Example 142](#_Toc423366100)

[3.1.8.2.2.2 Setting Compression and Extended Compression Flags 143](#_Toc423366101)

[3.1.8.2.3 Decompressing Data 144](#_Toc423366102)

[3.1.8.2.3.1 Data Decompression Example 146](#_Toc423366103)

[3.1.9 RDP 6.0 Bitmap Compression 148](#_Toc423366104)

[3.1.9.1 Bitmap Compression Techniques 148](#_Toc423366105)

[3.1.9.1.1 Splitting and Combining Color Planes 148](#_Toc423366106)

[3.1.9.1.2 Color Space Conversion 149](#_Toc423366107)

[3.1.9.1.3 Chroma Subsampling and Super-Sampling 150](#_Toc423366108)

[3.1.9.1.4 Color Loss Reduction 151](#_Toc423366109)

[3.1.9.2 Run-Length Encoding 152](#_Toc423366110)

[3.1.9.2.1 Encoding Run-Length Sequences 153](#_Toc423366111)

[3.1.9.2.2 Extra Long RUN Sequences 155](#_Toc423366112)

[3.1.9.2.3 Decoding Run-Length Sequences 156](#_Toc423366113)

[3.1.9.3 Compressing a Bitmap 158](#_Toc423366114)

[3.1.9.4 Decompressing a Bitmap 159](#_Toc423366115)

[3.2 Client Details 160](#_Toc423366116)

[3.2.1 Abstract Data Model 160](#_Toc423366117)

[3.2.1.1 Primary Drawing Order History 161](#_Toc423366118)

[3.2.1.2 Save Bitmap 161](#_Toc423366119)

[3.2.1.3 Bitmap Cache 161](#_Toc423366120)

[3.2.1.4 Persistent Bitmap Cache 161](#_Toc423366121)

[3.2.1.5 Persisted Bitmap Keys 161](#_Toc423366122)

[3.2.2 Timers 161](#_Toc423366123)

[3.2.3 Initialization 161](#_Toc423366124)

[3.2.4 Higher-Layer Triggered Events 161](#_Toc423366125)

[3.2.5 Processing Events and Sequencing Rules 162](#_Toc423366126)

[3.2.5.1 Drawing Orders 162](#_Toc423366127)

[3.2.5.1.1 Primary Drawing Orders 162](#_Toc423366128)

[3.2.5.1.1.1 Processing Primary Drawing Orders 162](#_Toc423366129)

[3.2.5.1.1.1.1 Processing of DstBlt 163](#_Toc423366130)

[3.2.5.1.1.1.2 Processing of MultiDstBlt 163](#_Toc423366131)

[3.2.5.1.1.1.3 Processing of PatBlt 163](#_Toc423366132)

[3.2.5.1.1.1.4 Processing of MultiPatBlt 163](#_Toc423366133)

[3.2.5.1.1.1.5 Processing of OpaqueRect 163](#_Toc423366134)

[3.2.5.1.1.1.6 Processing of MultiOpaqueRect 164](#_Toc423366135)

[3.2.5.1.1.1.7 Processing of ScrBlt 164](#_Toc423366136)

[3.2.5.1.1.1.8 Processing of MultiScrBlt 164](#_Toc423366137)

[3.2.5.1.1.1.9 Processing of MemBlt 164](#_Toc423366138)

[3.2.5.1.1.1.10 Processing of Mem3Blt 164](#_Toc423366139)

[3.2.5.1.1.1.11 Processing of LineTo 165](#_Toc423366140)

[3.2.5.1.1.1.12 Processing of SaveBitmap 165](#_Toc423366141)

[3.2.5.1.1.1.13 Processing of GlyphIndex 165](#_Toc423366142)

[3.2.5.1.1.1.14 Processing of FastIndex 165](#_Toc423366143)

[3.2.5.1.1.1.15 Processing of FastGlyph 165](#_Toc423366144)

[3.2.5.1.1.1.16 Processing of PolygonSC 166](#_Toc423366145)

[3.2.5.1.1.1.17 Processing of PolygonCB 166](#_Toc423366146)

[3.2.5.1.1.1.18 Processing of Polyline 166](#_Toc423366147)

[3.2.5.1.1.1.19 Processing of EllipseSC 166](#_Toc423366148)

[3.2.5.1.1.1.20 Processing of EllipseCB 166](#_Toc423366149)

[3.2.5.1.1.1.21 Processing of DrawNineGrid 166](#_Toc423366150)

[3.2.5.1.1.1.22 Processing of MultiDrawNineGrid 167](#_Toc423366151)

[3.2.5.1.2 Secondary Drawing Orders 167](#_Toc423366152)

[3.2.5.1.2.1 Processing Secondary Drawing Orders 167](#_Toc423366153)

[3.2.5.1.2.1.1 Processing of Cache Bitmap (Revision 1) 167](#_Toc423366154)

[3.2.5.1.2.1.2 Processing of Cache Bitmap (Revision 2) 167](#_Toc423366155)

[3.2.5.1.2.1.3 Processing of Cache Color Table 168](#_Toc423366156)

[3.2.5.1.2.1.4 Processing of Cache Glyph (Revision 1) 168](#_Toc423366157)

[3.2.5.1.2.1.5 Processing of Cache Glyph (Revision 2) 168](#_Toc423366158)

[3.2.5.1.2.1.6 Processing of Cache Brush 168](#_Toc423366159)

[3.2.5.1.2.1.7 Processing of Cache Bitmap (Revision 3) 169](#_Toc423366160)

[3.2.5.1.3 Alternate Secondary Drawing Orders 169](#_Toc423366161)

[3.2.5.1.3.1 Processing Alternate Secondary Drawing Orders 169](#_Toc423366162)

[3.2.5.1.3.1.1 Processing of Create Offscreen Bitmap 169](#_Toc423366163)

[3.2.5.1.3.1.2 Processing of Switch Surface 170](#_Toc423366164)

[3.2.5.1.3.1.3 Processing of Create NineGrid Bitmap 170](#_Toc423366165)

[3.2.5.1.3.1.4 Processing of Stream Bitmap Orders 170](#_Toc423366166)

[3.2.5.1.3.1.5 GDI+ Orders 170](#_Toc423366167)

[3.2.5.1.3.1.5.1 Processing of Draw GDI+ Cache Orders 170](#_Toc423366168)

[3.2.5.1.3.1.5.2 Processing of Draw GDI+ Orders 171](#_Toc423366169)

[3.2.5.2 Error Conditions 171](#_Toc423366170)

[3.2.5.2.1 Sending of Bitmap Cache Error PDU 171](#_Toc423366171)

[3.2.5.2.2 Sending of the Offscreen Bitmap Cache Error PDU 171](#_Toc423366172)

[3.2.5.2.3 Sending of the DrawNineGrid Cache Error PDU 171](#_Toc423366173)

[3.2.5.2.4 Sending of the GDI+ Error PDU 172](#_Toc423366174)

[3.2.6 Timer Events 172](#_Toc423366175)

[3.2.7 Other Local Events 172](#_Toc423366176)

[3.3 Server Details 172](#_Toc423366177)

[3.3.1 Abstract Data Model 172](#_Toc423366178)

[3.3.1.1 Cached Bitmap Keys 172](#_Toc423366179)

[3.3.1.2 Primary Drawing Order History 172](#_Toc423366180)

[3.3.1.3 Bitmap Cache Wait List 173](#_Toc423366181)

[3.3.2 Timers 173](#_Toc423366182)

[3.3.3 Initialization 173](#_Toc423366183)

[3.3.4 Higher-Layer Triggered Events 173](#_Toc423366184)

[3.3.5 Processing Events and Sequencing Rules 173](#_Toc423366185)

[3.3.5.1 Drawing Orders 173](#_Toc423366186)

[3.3.5.1.1 Primary Drawing Orders 174](#_Toc423366187)

[3.3.5.1.1.1 Construction of a Primary Drawing Order 174](#_Toc423366188)

[3.3.5.1.1.1.1 Construction of DstBlt 174](#_Toc423366189)

[3.3.5.1.1.1.2 Construction of MultiDstBlt 174](#_Toc423366190)

[3.3.5.1.1.1.3 Construction of PatBlt 174](#_Toc423366191)

[3.3.5.1.1.1.4 Construction of MultiPatBlt 175](#_Toc423366192)

[3.3.5.1.1.1.5 Construction of OpaqueRect 175](#_Toc423366193)

[3.3.5.1.1.1.6 Construction of MultiOpaqueRect 175](#_Toc423366194)

[3.3.5.1.1.1.7 Construction of ScrBlt 175](#_Toc423366195)

[3.3.5.1.1.1.8 Construction of MultiScrBlt 175](#_Toc423366196)

[3.3.5.1.1.1.9 Construction of MemBlt 176](#_Toc423366197)

[3.3.5.1.1.1.10 Construction of Mem3Blt 176](#_Toc423366198)

[3.3.5.1.1.1.11 Construction of LineTo 177](#_Toc423366199)

[3.3.5.1.1.1.12 Construction of SaveBitmap 177](#_Toc423366200)

[3.3.5.1.1.1.13 Construction of GlyphIndex 177](#_Toc423366201)

[3.3.5.1.1.1.14 Construction of FastIndex 177](#_Toc423366202)

[3.3.5.1.1.1.15 Construction of FastGlyph 178](#_Toc423366203)

[3.3.5.1.1.1.16 Construction of PolygonSC 178](#_Toc423366204)

[3.3.5.1.1.1.17 Construction of PolygonCB 178](#_Toc423366205)

[3.3.5.1.1.1.18 Construction of PolyLine 178](#_Toc423366206)

[3.3.5.1.1.1.19 Construction of EllipseSC 178](#_Toc423366207)

[3.3.5.1.1.1.20 Construction of EllipseCB 178](#_Toc423366208)

[3.3.5.1.1.1.21 Construction of DrawNineGrid 179](#_Toc423366209)

[3.3.5.1.1.1.22 Construction of MultiDrawNineGrid 179](#_Toc423366210)

[3.3.5.1.2 Secondary Drawing Orders 179](#_Toc423366211)

[3.3.5.1.2.1 Construction of Secondary Drawing Orders 179](#_Toc423366212)

[3.3.5.1.2.1.1 Construction of Cache Bitmap (Revision 1) 179](#_Toc423366213)

[3.3.5.1.2.1.2 Construction of Cache Bitmap (Revision 2) 180](#_Toc423366214)

[3.3.5.1.2.1.3 Construction of Cache Color Table 180](#_Toc423366215)

[3.3.5.1.2.1.4 Construction of Cache Glyph (Revision 1) 181](#_Toc423366216)

[3.3.5.1.2.1.5 Construction of Cache Glyph (Revision 2) 181](#_Toc423366217)

[3.3.5.1.2.1.6 Construction of Cache Brush 181](#_Toc423366218)

[3.3.5.1.2.1.7 Construction of Cache Bitmap (Revision 3) 182](#_Toc423366219)

[3.3.5.1.3 Alternate Secondary Drawing Orders 182](#_Toc423366220)

[3.3.5.1.3.1 Construction of Alternate Secondary Drawing Orders 182](#_Toc423366221)

[3.3.5.1.3.1.1 Construction of Create Offscreen Bitmap 182](#_Toc423366222)

[3.3.5.1.3.1.2 Construction of Switch Surface 183](#_Toc423366223)

[3.3.5.1.3.1.3 Construction of Create NineGrid Bitmap 183](#_Toc423366224)

[3.3.5.1.3.1.4 Construction of Stream Bitmap Orders 183](#_Toc423366225)

[3.3.5.1.3.1.5 GDI+ Orders 184](#_Toc423366226)

[3.3.5.1.3.1.5.1 Construction of Draw GDI+ Cache Orders 184](#_Toc423366227)

[3.3.5.1.3.1.5.2 Construction of Draw GDI+ Orders 184](#_Toc423366228)

[3.3.5.2 Error Conditions 184](#_Toc423366229)

[3.3.5.2.1 Processing of Bitmap Cache Error PDU 184](#_Toc423366230)

[3.3.5.2.2 Processing of the Offscreen Bitmap Cache Error PDU 184](#_Toc423366231)

[3.3.5.2.3 Processing of the DrawNineGrid Cache Error PDU 185](#_Toc423366232)

[3.3.5.2.4 Processing of the GDI+ Error PDU 185](#_Toc423366233)

[3.3.6 Timer Events 185](#_Toc423366234)

[3.3.7 Other Local Events 185](#_Toc423366235)

[4 Protocol Examples 186](#_Toc423366236)

[4.1 Annotated Primary Drawing Orders 186](#_Toc423366237)

[4.1.1 DstBlt 186](#_Toc423366238)

[4.1.2 MultiDstBlt 186](#_Toc423366239)

[4.1.3 PatBlt 187](#_Toc423366240)

[4.1.4 MultiPatBlt 188](#_Toc423366241)

[4.1.5 OpaqueRect 189](#_Toc423366242)

[4.1.6 MultiOpaqueRect 189](#_Toc423366243)

[4.1.7 ScrBlt 190](#_Toc423366244)

[4.1.8 MultiScrBlt 191](#_Toc423366245)

[4.1.9 MemBlt 193](#_Toc423366246)

[4.1.10 Mem3Blt 194](#_Toc423366247)

[4.1.11 LineTo 195](#_Toc423366248)

[4.1.12 SaveBitmap 196](#_Toc423366249)

[4.1.13 GlyphIndex 196](#_Toc423366250)

[4.1.13.1 Example 1 196](#_Toc423366251)

[4.1.13.2 Example 2 197](#_Toc423366252)

[4.1.14 FastIndex 198](#_Toc423366253)

[4.1.14.1 Example 1 198](#_Toc423366254)

[4.1.14.2 Example 2 199](#_Toc423366255)

[4.1.15 FastGlyph 200](#_Toc423366256)

[4.1.16 PolygonSC 201](#_Toc423366257)

[4.1.17 PolygonCB 204](#_Toc423366258)

[4.1.18 Polyline 205](#_Toc423366259)

[4.1.19 EllipseSC 207](#_Toc423366260)

[4.1.20 EllipseCB 208](#_Toc423366261)

[4.1.21 DrawNineGrid 209](#_Toc423366262)

[4.1.22 MultiDrawNineGrid 210](#_Toc423366263)

[4.2 Annotated Secondary Drawing Orders 211](#_Toc423366264)

[4.2.1 Cache Bitmap (Revision 1) 211](#_Toc423366265)

[4.2.2 Cache Bitmap (Revision 2) 211](#_Toc423366266)

[4.2.3 Cache Color Table 214](#_Toc423366267)

[4.2.4 Cache Glyph (Revision 1) 219](#_Toc423366268)

[4.2.5 Cache Glyph (Revision 2) 220](#_Toc423366269)

[4.2.6 Cache Brush 221](#_Toc423366270)

[4.2.7 Cache Bitmap (Revision 3) 222](#_Toc423366271)

[4.3 Annotated Alternate Secondary Drawing Orders 223](#_Toc423366272)

[4.3.1 Create Offscreen Bitmap 223](#_Toc423366273)

[4.3.2 Switch Surface 223](#_Toc423366274)

[4.3.3 Create NineGrid Bitmap 223](#_Toc423366275)

[4.3.4 Stream Bitmap First 224](#_Toc423366276)

[4.3.5 Stream Bitmap Next 225](#_Toc423366277)

[4.3.6 Draw GDI+ Cache First 225](#_Toc423366278)

[4.3.6.1 Example 1 225](#_Toc423366279)

[4.3.6.2 Example 2 229](#_Toc423366280)

[4.3.7 Draw GDI+ Cache Next 230](#_Toc423366281)

[4.3.8 Draw GDI+ Cache End 234](#_Toc423366282)

[4.3.9 Draw GDI+ First 234](#_Toc423366283)

[4.3.10 Draw GDI+ Next 238](#_Toc423366284)

[4.3.11 Draw GDI+ End 242](#_Toc423366285)

[4.4 NineGrid Examples 243](#_Toc423366286)

[4.5 Save Bitmap Example 245](#_Toc423366287)

[4.6 Glyph Image Data 247](#_Toc423366288)

[4.6.1 "d" Character 247](#_Toc423366289)

[4.6.2 "p" Character 248](#_Toc423366290)

[4.7 RDP 6.1 Bulk Compression Example 248](#_Toc423366291)

[4.8 Error Conditions 249](#_Toc423366292)

[4.8.1 Bitmap Cache Error PDU 249](#_Toc423366293)

[4.8.2 Offscreen Bitmap Cache Error PDU 250](#_Toc423366294)

[4.8.3 DrawNineGrid Cache Error PDU 251](#_Toc423366295)

[4.8.4 GDI+ Error PDU 252](#_Toc423366296)

[5 Security 253](#_Toc423366297)

[5.1 Security Considerations for Implementers 253](#_Toc423366298)

[5.2 Index of Security Parameters 253](#_Toc423366299)

[6 Appendix A: Product Behavior 254](#_Toc423366300)

[7 Change Tracking 255](#_Toc423366301)

[8 Index 257](#_Toc423366302)

# Introduction

The Remote Desktop Protocol: Graphics Devices Interfaces (GDI) Acceleration Extension is an extension to the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting (as specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf)). The aim of the Remote Desktop Protocol: GDI Acceleration Extension is to reduce the bandwidth associated with graphics remoting by encoding the drawing operations that produce an image instead of encoding the actual image.

Sections 1.8, 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]](http://go.microsoft.com/fwlink/?LinkId=90317). 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.

## Glossary

The following terms are specific to this document:

**ANSI character**: An 8-bit Windows-1252 character set unit.

**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.

**ASN.1**: Abstract Syntax Notation One. ASN.1 is used to describe Kerberos datagrams as a sequence of components, sent in messages. ASN.1 is described in the following specifications: [[ITUX660]](http://go.microsoft.com/fwlink/?LinkId=89922) for general procedures; [[ITUX680]](http://go.microsoft.com/fwlink/?LinkId=89923) for syntax specification, and [[ITUX690]](http://go.microsoft.com/fwlink/?LinkId=89924) for the Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER) encoding rules.

**AYCoCg**: A color space in which each color is represented as a quad (A, Y, Co, Cg), where A represents the alpha (transparency) component, Y represents the luma (intensity) component, and Co and Cg represent the two chrominance (color) components orange and green, respectively.

**brush**: An 8-by-8-pixel bitmap that is repeated horizontally and vertically to fill an area.

**client**: A computer on which the remote procedure call (RPC) client is executing.

**color plane**: A two-dimensional surface containing a collection of values that represent a single component of the [**ARGB**](#gt_a99d8c7d-4340-403d-b965-1f1e0b7defb1) or [**AYCoCg**](#gt_74e654be-4f30-46ff-982a-e7c497e3152b) color space.

**color space**: Any method of representing colors for printing or electronic display.

**Packed Encoding Rules (PER)**: A set of encoding rules for [**ASN.1**](#gt_2ae47996-2ca2-4ca7-9e80-8ba36f62051a) notation, specified in [[ITUX691]](http://go.microsoft.com/fwlink/?LinkId=192078). These rules enable the identification, extraction, and decoding of data structures.

**protocol data unit (PDU)**: Information that is delivered as a unit among peer entities of a network and that may contain control information, address information, or data. For more information on remote procedure call (RPC)-specific PDUs, see [[C706]](http://go.microsoft.com/fwlink/?LinkId=89824) section 12.

**red-green-blue (RGB)**: A color model that describes color information in terms of the red (R), green (G), and blue (B) intensities in a color.

**Reverse Polish Notation (RPN)**: A mathematical notation wherein each operator follows all of its operands. Also known as postfix notation.

**screen tearing**: A phenomenon in video rendering where a newly rendered frame partially overlaps with a previously rendered frame, creating a torn look as graphical objects do not line up.

**server**: A computer on which the remote procedure call (RPC) server is executing.

**Unicode character**: Unless otherwise specified, a 16-bit UTF-16 code unit.

**YCoCg**: A color space wherein each color is represented as a triple (Y, Co, Cg), where Y represents the luma (intensity) component, and Co and Cg represent the two chrominance (color) components orange and green, respectively.

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

## References

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

### Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[MS-EMFPLUS] Microsoft Corporation, "[Enhanced Metafile Format Plus Extensions](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf)".

[MS-RDPBCGR] Microsoft Corporation, "[Remote Desktop Protocol: Basic Connectivity and Graphics Remoting](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf)".

[MS-RDPEDC] Microsoft Corporation, "[Remote Desktop Protocol: Desktop Composition Virtual Channel Extension](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPEDC%5d.pdf)".

[MS-RDPERP] Microsoft Corporation, "[Remote Desktop Protocol: Remote Programs Virtual Channel Extension](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPERP%5d.pdf)".

[PETZOLD] Petzold, C., "Programming Windows, Fifth Edition", Microsoft Press, 1998, ISBN: 157231995X.

[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](http://go.microsoft.com/fwlink/?LinkId=90317)

[T123] ITU-T, "Network-Specific Data Protocol Stacks for Multimedia Conferencing", Recommendation T.123, May 1999, [http://www.itu.int/rec/T-REC-T.123/en](http://go.microsoft.com/fwlink/?LinkId=90541)

**Note** There is a charge to download the specification.

[T125] ITU-T, "Multipoint Communication Service Protocol Specification", Recommendation T.125, February 1998, [http://www.itu.int/rec/T-REC-T.125-199802-I/en](http://go.microsoft.com/fwlink/?LinkId=90543)

**Note** There is a charge to download the specification.

[T128] ITU-T, "Multipoint Application Sharing", Recommendation T.128, February 1998, [http://www.itu.int/rec/T-REC-T.128-199802-S/en](http://go.microsoft.com/fwlink/?LinkId=90544)

**Note** There is a charge to download the specification.

[X224] ITU-T, "Information technology - Open Systems Interconnection - Protocol for Providing the Connection-Mode Transport Service", Recommendation X.224, November 1995, [http://www.itu.int/rec/T-REC-X.224-199511-I/en](http://go.microsoft.com/fwlink/?LinkId=90588)

**Note** There is a charge to download the specification.

[YUAN] Yuan, F., "Windows Graphics Programming - Win32 GDI and DirectDraw", Prentice Hall PTR, 2000, ISBN: 0130869856.

### Informative References

[MSDN-ABLEND] Microsoft Corporation, "AlphaBlend function", [http://msdn.microsoft.com/en-us/library/dd183351.aspx](http://go.microsoft.com/fwlink/?LinkId=89948)

[MSDN-BitBlt] Microsoft Corporation, "BitBlt function", [http://msdn.microsoft.com/en-us/library/dd183370.aspx](http://go.microsoft.com/fwlink/?LinkId=89958)

[MSDN-CW] Microsoft Corporation, "Character Widths", [http://msdn.microsoft.com/en-us/library/dd183418.aspx](http://go.microsoft.com/fwlink/?LinkId=196874)

[MSDN-SWH] Microsoft Corporation, "String Widths and Heights", [http://msdn.microsoft.com/en-us/library/dd145122.aspx](http://go.microsoft.com/fwlink/?LinkId=196875)

[MSDN-TransparentBlt] Microsoft Corporation, "TransparentBlt function", [http://msdn.microsoft.com/en-us/library/dd145141.aspx](http://go.microsoft.com/fwlink/?LinkId=90146)

[NINEGRID] Microsoft Corporation, "Using Nine-Grid Rendering", [http://msdn.microsoft.com/en-us/library/bb189722.aspx](http://go.microsoft.com/fwlink/?LinkId=90233)

[SAYOOD] Sayood, K., "Lossless Compression Handbook, First Edition", Academic Press, August 2002, ISBN: 0126208611.

## Overview

The Remote Desktop Protocol: GDI Acceleration Extension reduces the bandwidth associated with graphics remoting. The following sections provide an overview of the major components of this protocol and how bandwidth reduction is achieved.

### Accelerated Graphics

The remoting of graphics images (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) sections 2.2.9.1.1.3.1.2 and 2.2.9.1.2.1.2) is accomplished by continuously sending updated bitmap images from [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) to [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506). Even though these bitmaps may be compressed, it is still not a bandwidth-efficient mechanism to employ, especially when dealing with graphics-intensive applications that refresh regularly.

The Remote Desktop Protocol: GDI Acceleration Extension aims to reduce the bandwidth associated with graphics remoting by encoding the drawing operations that produce an image instead of encoding the actual image.

For example, instead of sending the bitmap image of a filled rectangle from server to client, an order to render a rectangle at coordinate (X, Y) with a given width, height, and fill color is sent to the client. The client then executes the drawing order to produce the intended graphics result.

In addition to defining how to encode common drawing operations, the Remote Desktop Protocol: GDI Acceleration Extension also facilitates the use of caches to store drawing primitives such as bitmaps, color tables, and characters. The effective use of caching techniques helps to reduce wire traffic by ensuring that items used in multiple drawing operations are sent only once from server to client (retransmission of these items for use in conjunction with future drawing operations is not required after the item has been cached on the client).

#### Caches

The Remote Desktop Protocol: GDI Acceleration Extension defines a number of caches that may be leveraged by clients and servers:

* Bitmap Cache: Stores bitmap images.
* Color Table Cache: Stores color palettes.
* Glyph Cache: Stores character images.
* Fragment Cache: Stores collections of glyphs.
* Brush Cache: Stores 8-by-8-pixel bitmaps used to fill regions.
* Offscreen Bitmap Cache: Stores writable bitmaps.
* GDI+ Caches: Used to cache GDI+ 1.1 primitives:
  + Graphics Cache
  + Brush Cache
  + Pen Cache
  + Image Attributes Cache
  + Image Cache
* NineGrid Bitmap Cache: Stores NineGrid-compliant bitmaps. For more information about nine-grid bitmaps, see [[NINEGRID]](http://go.microsoft.com/fwlink/?LinkId=90233).

All of the caches (except the bitmap cache) are memory-based and are not persisted across connections. The use of caches is optional and is specified through the use of capability sets.

Encoded drawing operations that use cached items refer to these items by specifying the cache entry in which the item is stored (if there are multiple caches, the cache ID also needs to be specified). This implies that an item must first be cached (by the server and client) before any drawing operations that reference it can be sent.

For example, a server may instruct a client to cache a particular [**brush**](#gt_651044e0-b864-4fdb-aba6-f4cd25b0b3c8) pattern in the Brush Cache, for example, at index location 23. Then, in a subsequent drawing operation that involves a brush pattern, the server may instruct the client to use the brush pattern stored in the Brush Cache at index location 23.

Entries can be evicted from a cache if room needs to be made for new items. The server determines what entries should be evicted and then instructs the client to perform the eviction. In this way, the client and server caches remain in sync.

#### Drawing Orders

Drawing orders are used to perform the following operations:

* Transport bitmap data
* Encode graphics rendering primitives
* Manipulate data caches
* Manage rendering surfaces
* Support application remoting ([[MS-RDPERP]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPERP%5d.pdf) section 1.3)
* Support desktop composition ([[MS-RDPEDC]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPEDC%5d.pdf) section 1.3)

There are three classes of drawing orders:

* Primary
* Secondary
* Alternate secondary

These three classes of drawing orders are described in sections [1.3.1.2.1](#Section_A8A67FE14117453D9EA2AB67BC1C10EF) through to [1.3.1.2.3](#Section_6255B5ECF77E49F0A4BF77B6C748CD64).

##### Primary Drawing Orders

Primary orders are mainly used to encode drawing operations. Each primary order is organized into a set of fields to which field-compression algorithms are applied. These algorithms are designed to eliminate sending a field if it has not changed since the last time the order was sent and to reduce the size of the field encoding for certain field types when they can be represented by smaller sized data.

The following are the broad categories of drawing operations that primary orders encode.

* Combining bitmap patterns using raster operations (called a "blit").
* Drawing graphic objects:
  + Rectangles
  + Lines
  + Polygons
  + Ellipses
* Displaying text fragments
* Transferring portions of the screen area from one point to another
* Temporarily saving obscured regions of the screen
* Rendering cached bitmaps
* Rendering cached NineGrid bitmaps

A subset of the primary drawing operations requires rendering data to be present in a specific cache. For example, to render a NineGrid bitmap, it must be present in the NineGrid cache. This implies that some of the primary drawing orders have a dependency on the secondary orders to first transmit the data necessary to complete a drawing operation to the client.

If a given primary drawing order cannot be constructed, the server must send screen bitmap data in a Bitmap Update (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) sections 2.2.9.1.1.3.1.2 and 2.2.9.1.2.1.2) for the screen area affected by the primary drawing order. The use of the primary drawing orders is specified by using capability sets.

##### Secondary Drawing Orders

Secondary drawing orders are primarily used to manage the addition and removal of items from the following four caches:

* Bitmap Cache
* Color Table Cache
* Glyph Cache
* Brush Cache

Use of the secondary orders is specified through capabilities. If the existence of one of the previous four caches is specified, the secondary orders associated with that cache are presumed to be supported.

##### Alternate Secondary Drawing Orders

Alternate secondary drawing orders are used to extend the existing set of Primary and Secondary RDP drawing orders. The alternate secondary drawing orders described in this document are used to manage the Offscreen and NineGrid Bitmap Caches and to transport opaque EMF+ records that contain GDI+ 1.1 primitives (as specified in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3.1).

Using these orders, a bitmap can be created in the Offscreen Bitmap Cache, and then set as the target drawing surface on the client. Alternate secondary drawing orders also enable the creation and efficient population of NineGrid bitmaps in the NineGrid bitmap cache. Besides creating and updating entries in these two caches, the alternate secondary orders also support deletion of entries.

GDI+ 1.1 remoting is enabled through the use of alternate secondary drawing orders. In this scenario, the orders are used to transport opaque GDI+ 1.1 records between endpoints, in addition to caching instructions to store GDI+ 1.1 primitives.

Other alternate secondary drawing orders added by RDP extensions are used to support remoting applications ([[MS-RDPERP]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPERP%5d.pdf) section 2.2.1.3) and desktop composition ([[MS-RDPEDC]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPEDC%5d.pdf) section 2.2).

Use of the alternate secondary orders is specified through capabilities.

#### Error Conditions

The Remote Desktop Protocol: GDI Acceleration Extension enables the client to inform the server of errors in the following areas:

* Bitmap caching
* Offscreen bitmap caching
* NineGrid caching
* GDI+ 1.1 remoting

If a client encounters an error in any one of these areas, it can send an appropriate [**protocol data unit (PDU)**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) to the server to notify it of this issue and ensure that appropriate action is taken, such as disabling features within the area.

### Bulk Data Compression

#### RDP 6.0

The Remote Desktop Protocol: GDI Acceleration Extension builds on the bulk data compression mechanisms described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8 to create a compressor that employs Huffman encoding algorithms [HUFFCODE]. This new bulk compression technique (known as "RDP 6.0 Bulk Compression") produces higher compression ratios, and as a result, improves the overall bandwidth consumption of server-to-client traffic (it is not used for client-to-server traffic).

#### RDP 6.1

Remote Desktop Protocol: GDI Acceleration Extension supports an efficient large history bulk data (pre)compressor that is serially chained with the pre-existing bit-level RDP 5.0 bulk compressor (described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8). The RDP 6.1 bulk compression technique produces better compression ratios than the RDP 6.0 bulk compressor, but also consumes significantly more memory, which may affect scalability of multi-user servers. This compressor is only used for server-to-client traffic (it is not used for client-to-server traffic).

## Relationship to Other Protocols

This protocol extends the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting (as specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf)) by adding advanced drawing order capabilities and compression techniques.

## Prerequisites/Preconditions

All the packets defined by this protocol are tunneled within the RDP transport ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.1). Hence, this protocol only operates after the RDP transport has been fully established.

## Applicability Statement

This protocol is applicable in situations in which it is necessary to optimize the bandwidth required for graphics remoting. The advanced drawing orders specified in this document enable drawing operations to be sent using compact representations, eliminating the need to send bitmaps in many common drawing operations.

## Versioning and Capability Negotiation

This protocol builds on the basic Remote Desktop Protocol. The features provided by this extension are selected during the Capabilities Exchange Phase of the RDP Connection Sequence (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 1.3.1.1). In effect, this extension merely expands the set of capabilities used by the base RDP. (RDP versioning and capability exchange is described in [MS-RDPBCGR] section 1.7.)

## Vendor-Extensible Fields

None.

## Standards Assignments

None.

# Messages

## Transport

This protocol is an extension to the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting, and all packets are tunneled within the RDP transport ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.1).

## Message Syntax

All multiple-byte fields within a message MUST be marshaled in little-endian byte order, unless otherwise specified.

Version 2 MCS Encoding Rules (defined in [[T125]](http://go.microsoft.com/fwlink/?LinkId=90543) section 9) are used in encoding MCS structures defined in [T125].

### Capability Sets

#### Color Table Cache Capability Set (TS\_COLORTABLE\_CAPABILITYSET)

The TS\_COLORTABLE\_CAPABILITYSET structure is an unused capability set that advertises the size of the color table cache used in conjunction with the Cache Color Table Secondary Drawing Order (see section [2.2.2.2.1.2.4](#Section_49fad30181714f38a68a7394baeb7b15)) and is based on the capability set in [[T128]](http://go.microsoft.com/fwlink/?LinkId=90544) section 8.2.8. This capability is sent by both client and server.

Instead of being specified by the Color Table Cache Capability Set, the existence of color table caching is tied to support for the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing orders. If support for these orders is advertised in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the existence of a color table cache with entries for six palettes is implied when palettized color is being used.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| capabilitySetType | | | | | | | | | | | | | | | | lengthCapability | | | | | | | | | | | | | | | |
| colorTableCacheSize | | | | | | | | | | | | | | | | pad2octets | | | | | | | | | | | | | | | |

**capabilitySetType (2 bytes):** A 16-bit, unsigned integer. The type of the capability set. This field MUST be set to CAPSTYPE\_COLORCACHE (0x000A).

**lengthCapability (2 bytes):** A 16-bit, unsigned integer. The length, in bytes, of the capability data, including the size of the **capabilitySetType** and **lengthCapability** fields.

**colorTableCacheSize (2 bytes):** A 16-bit, unsigned integer. The number of entries in the color table cache (each entry stores a color table). This value MUST be ignored during capability exchange and is assumed to be 0x0006.

**pad2octets (2 bytes):** A 16-bit, unsigned integer used as padding. Values in this field are arbitrary and MUST be ignored.

#### DrawNineGrid Cache Capability Set (TS\_DRAW\_NINEGRID\_CAPABILITYSET)

The TS\_DRAW\_NINEGRID\_CAPABILITYSET structure is used to advertise support for NineGrid bitmap caching and rendering (see sections [2.2.2.2.1.1.2.21](#Section_44bd6a68a5b74a38980706ba4ab4a72a), [2.2.2.2.1.1.2.22](#Section_101fecdc3ad841c6b7b670d8efd4ad70), and [2.2.2.2.1.3.4](#Section_6ac5eb820c30405490e883797f24bc38)). This capability set is sent only from client to server.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| capabilitySetType | | | | | | | | | | | | | | | | lengthCapability | | | | | | | | | | | | | | | |
| drawNineGridSupportLevel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| drawNineGridCacheSize | | | | | | | | | | | | | | | | drawNineGridCacheEntries | | | | | | | | | | | | | | | |

**capabilitySetType (2 bytes):** A 16-bit, unsigned integer. The type of the capability set. This field MUST be set to CAPSTYPE\_DRAWNINEGRIDCACHE (0x0015).

**lengthCapability (2 bytes):** A 16-bit, unsigned integer. The length, in bytes, of the capability data, including the size of the **capabilitySetType** and **lengthCapability** fields.

**drawNineGridSupportLevel (4 bytes):** A 32-bit, unsigned integer. The level of support for NineGrid drawing. This field MUST be set to one of the following values.

| Value | Meaning |
| --- | --- |
| DRAW\_NINEGRID\_NO\_SUPPORT  0x00000000 | NineGrid bitmap caching and rendering is not supported. |
| DRAW\_NINEGRID\_SUPPORTED  0x00000001 | Revision 1 NineGrid bitmap caching and rendering is supported. The Revision 1 versions of the stream bitmap alternate secondary orders (see section [2.2.2.2.1.3.5](#Section_5d03486bb73b463cb77330691af9dffc)) MUST be used to send the NineGrid bitmap from server to client. |
| DRAW\_NINEGRID\_SUPPORTED\_REV2  0x00000002 | Revision 2 NineGrid bitmap caching and rendering is supported. The Revision 2 versions of the stream bitmap alternate secondary orders (see section 2.2.2.2.1.3.5) MUST be used to send the NineGrid bitmap from server to client. |

**drawNineGridCacheSize (2 bytes):** A 16-bit, unsigned integer. The maximum size of the NineGrid Bitmap Cache. The largest size allowed by current RDP servers is 2,560 kilobytes.

**drawNineGridCacheEntries (2 bytes):** A 16-bit, unsigned integer. The maximum number of entries allowed in the NineGrid Bitmap Cache. The maximum number of entries allowed by current RDP servers is 256.

#### Draw GDI+ Capability Set (TS\_DRAW\_GDIPLUS\_CAPABILITYSET)

The TS\_DRAW\_GDIPLUS\_CAPABILITYSET structure is used to advertise the level of GDI+ 1.1 rendering and caching support and the GDI+ cache configuration. This capability is sent by both client and server. However, only the client initializes the **GdipVersion**, **GdipCacheEntries**, **GdipCacheChunkSize**, and **GdipImageCacheProperties** fields. The server uses this capability set only to advertise its level of GDI+ 1.1 rendering and caching support.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| capabilitySetType | | | | | | | | | | | | | | | | lengthCapability | | | | | | | | | | | | | | | |
| drawGDIPlusSupportLevel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GdipVersion | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| drawGdiplusCacheLevel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GdipCacheEntries | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | GdipCacheChunkSize | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | GdipImageCacheProperties | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**capabilitySetType (2 bytes):** A 16-bit, unsigned integer. The type of the capability set. This field MUST be set to CAPSTYPE\_DRAWGDIPLUS (0x0016).

**lengthCapability (2 bytes):** A 16-bit, unsigned integer. The length, in bytes, of the capability data, including the size of the **capabilitySetType** and **lengthCapability** fields.

**drawGDIPlusSupportLevel (4 bytes):** A 32-bit, unsigned integer. The level of support for GDI+ 1.1 remoting. This field MUST be set to one of the following values.

| Value | Meaning |
| --- | --- |
| TS\_DRAW\_GDIPLUS\_DEFAULT  0x00000000 | GDI+ 1.1 is not supported. |
| TS\_DRAW\_GDIPLUS\_SUPPORTED  0x00000001 | GDI+ 1.1 is supported. |

**GdipVersion (4 bytes):** A 32-bit, unsigned integer. The build number of the underlying GDI+ 1.1 subsystem. Only the client-to-server instance of the GDI+ Capability Set MUST contain a valid value for this field.

**drawGdiplusCacheLevel (4 bytes):** A 32-bit, unsigned integer. The level of support for the caching of GDI+ 1.1 rendering primitives. This field MUST be set to one of the following values.

| Value | Meaning |
| --- | --- |
| TS\_DRAW\_GDIPLUS\_CACHE\_LEVEL\_DEFAULT  0x00000000 | Caching of GDI+ 1.1 rendering primitives is not supported. |
| TS\_DRAW\_GDIPLUS\_CACHE\_LEVEL\_ONE  0x00000001 | Caching of GDI+ 1.1 rendering primitives is supported. |

**GdipCacheEntries (10 bytes):** A [GDI+ Cache Entries (section 2.2.1.3.1)](#Section_f7328112a3a445ea9022a28033912273) structure that specifies the total number of entries within the GDI+ Graphics, Pen, Brush, Image, and Image Attributes caches. Only the client-to-server instance of the GDI+ Capability Set MUST contain a valid value for this field.

**GdipCacheChunkSize (8 bytes):** A [GDI+ Cache Chunk Size](#Section_cfe6022291854ab4ab06b0ec6e2d174e) structure that specifies the size of individual entries in the GDI+ Graphics, Brush, Pen, and Image Attributes caches. Only the client-to-server instance of the GDI+ Capability Set MUST contain a valid value for this field.

**GdipImageCacheProperties (6 bytes):** A [GDI+ Image Cache Properties](#Section_73c1767cf10e4361b980f39e3c44f895) structure that contains sizing information for the GDI+ Image cache. Only the client-to-server instance of the GDI+ Capability Set MUST contain a valid value for this field.

##### GDI+ Cache Entries (TS\_GDIPLUS\_CACHE\_ENTRIES)

The TS\_GDIPLUS\_CACHE\_ENTRIES structure specifies the total number of cache entries for the GDI+ Graphics, Brush, Pen, Image, and Image Attributes caches.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| GdipGraphicsCacheEntries | | | | | | | | | | | | | | | | GdipBrushCacheEntries | | | | | | | | | | | | | | | |
| GdipPenCacheEntries | | | | | | | | | | | | | | | | GdipImageCacheEntries | | | | | | | | | | | | | | | |
| GdipImageAttributesCacheEntries | | | | | | | | | | | | | | | |

**GdipGraphicsCacheEntries (2 bytes):** A 16-bit, unsigned integer. The total number of entries allowed in the GDI+ Graphics cache. The maximum allowed value is 10 entries.

**GdipBrushCacheEntries (2 bytes):** A 16-bit, unsigned integer. The total number of entries allowed in the GDI+ Brush cache. The maximum allowed value is 5 entries.

**GdipPenCacheEntries (2 bytes):** A 16-bit, unsigned integer. The total number of entries allowed in the GDI+ Pen cache. The maximum allowed value is 5 entries.

**GdipImageCacheEntries (2 bytes):** A 16-bit, unsigned integer. The total number of entries allowed in the GDI+ Image cache. The maximum allowed value is 10 entries.

**GdipImageAttributesCacheEntries (2 bytes):** A 16-bit, unsigned integer. The total number of entries allowed in the GDI+ Image Attributes cache. The maximum allowed value is 2 entries.

##### GDI+ Cache Chunk Size (TS\_GDIPLUS\_CACHE\_CHUNK\_SIZE)

The TS\_GDIPLUS\_CACHE\_CHUNK\_SIZE structure specifies the maximum size of individual entries in the GDI+ Graphics, Brush, Pen, and Image Attributes caches.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| GdipGraphicsCacheChunkSize | | | | | | | | | | | | | | | | GdipObjectBrushCacheChunkSize | | | | | | | | | | | | | | | |
| GdipObjectPenCacheChunkSize | | | | | | | | | | | | | | | | GdipObjectImageAttributesCacheChunkSize | | | | | | | | | | | | | | | |

**GdipGraphicsCacheChunkSize (2 bytes):** A 16-bit, unsigned integer. The maximum size in bytes of a GDI+ Graphics cache entry. The maximum allowed value is 512 bytes.

**GdipObjectBrushCacheChunkSize (2 bytes):** A 16-bit, unsigned integer. The maximum size in bytes of a GDI+ Brush cache entry. The maximum allowed value is 2,048 bytes.

**GdipObjectPenCacheChunkSize (2 bytes):** A 16-bit, unsigned integer. The maximum size in bytes of a GDI+ Pen cache entry. The maximum allowed value is 1,024 bytes.

**GdipObjectImageAttributesCacheChunkSize (2 bytes):** A 16-bit, unsigned integer. The maximum size in bytes of a GDI+ Image Attributes cache entry. The maximum allowed value is 64 bytes.

##### GDI+ Image Cache Properties (TS\_GDIPLUS\_IMAGE\_CACHE\_PROPERTIES)

The TS\_GDIPLUS\_IMAGE\_CACHE\_PROPERTIES structure contains sizing information for the GDI+ Image cache.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| GdipObjectImageCacheChunkSize | | | | | | | | | | | | | | | | GdipObjectImageCacheTotalSize | | | | | | | | | | | | | | | |
| GdipObjectImageCacheMaxSize | | | | | | | | | | | | | | | |

**GdipObjectImageCacheChunkSize (2 bytes):** A 16-bit, unsigned integer. The maximum size in bytes of a chunk in the GDI+ Image cache. The maximum allowed value is 4,096 bytes.

**GdipObjectImageCacheTotalSize (2 bytes):** A 16-bit, unsigned integer. The total number of chunks in the GDI+ Image cache. The maximum allowed value is 256 chunks.

**GdipObjectImageCacheMaxSize (2 bytes):** A 16-bit, unsigned integer. The total number of chunks that can be used by an entry in the GDI+ Image cache. The maximum allowed value is 128 chunks.

### Accelerated Graphics

#### Orders Update (TS\_UPDATE\_ORDERS\_PDU\_DATA)

The TS\_UPDATE\_ORDERS\_PDU\_DATA structure contains primary, secondary, and alternate secondary drawing orders aligned on byte boundaries. This structure conforms to the layout of a Slow Path Graphics Update (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.1.3.1) and is encapsulated within a Graphics Update PDU (see [MS-RDPBCGR] section 2.2.9.1.1.3.1.1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| shareDataHeader (18 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | updateType | | | | | | | | | | | | | | | |
| pad2OctetsA | | | | | | | | | | | | | | | | numberOrders | | | | | | | | | | | | | | | |
| pad2OctetsB | | | | | | | | | | | | | | | | orderData (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**shareDataHeader (18 bytes):** Share Data Header (see [MS-RDPBCGR], section 2.2.8.1.1.1.2) containing information about the packet. The type subfield of the pduType field of the Share Control Header (section 2.2.8.1.1.1.1) MUST be set to PDUTYPE\_DATAPDU (7). The pduType2 field of the Share Data Header MUST be set to PDUTYPE2\_UPDATE (2).

**updateType (2 bytes):** A 16-bit, unsigned integer. The field contains the graphics update type. This field MUST be set to UPDATETYPE\_ORDERS (0x0000).

**pad2OctetsA (2 bytes):** A 16-bit, unsigned integer used as a padding field. Values in this field are arbitrary and MUST be ignored.

**numberOrders (2 bytes):** A 16-bit, unsigned integer. The number of [Drawing Order (section 2.2.2.1.1)](#Section_762f3dd36a76485786a4a3ca951ca3a6) structures contained in the **orderData** field.

**pad2OctetsB (2 bytes):** A 16-bit, unsigned integer used as a padding field. Values in this field are arbitrary and MUST be ignored.

**orderData (variable):** A variable-sized array of Drawing Order (section 2.2.2.1.1) structures packed on byte boundaries. Each structure contains a primary, secondary, or alternate secondary drawing order. The **controlFlags** field of the Drawing Order identifies the type of drawing order.

##### Drawing Order (DRAWING\_ORDER)

The DRAWING\_ORDER structure encapsulated by the [Orders Update (section 2.2.2.1)](#Section_ef7c27f5097746fea50ef9a51d6ab90b) is identical to the DRAWING\_ORDER structure used in conjunction with the [Fast-Path Orders Update (section 2.2.2.2)](#Section_7f53bb2892e94cbea15f8553c902e3c4).

#### Fast-Path Orders Update (TS\_FP\_UPDATE\_ORDERS)

The TS\_FP\_UPDATE\_ORDERS structure contains primary, secondary, and alternate secondary drawing orders aligned on byte boundaries. This structure conforms to the layout of a Fast-Path Update (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.2.1) and is encapsulated within a Fast-Path Update PDU (see [MS-RDPBCGR] section 2.2.9.1.2.1.1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| updateHeader | | | | | | | | compressionFlags (optional) | | | | | | | | size | | | | | | | | | | | | | | | |
| numberOrders | | | | | | | | | | | | | | | | orderData (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**updateHeader (1 byte):** An 8-bit, unsigned integer. The format of this field is the same as the **updateHeader** byte field described in the Fast-Path Update structure (see [MS-RDPBCGR] section 2.2.9.1.2.1). The **updateCode** bitfield (4 bits in size) MUST be set to FASTPATH\_UPDATETYPE\_ORDERS (0x0).

**compressionFlags (1 byte):** An 8-bit, unsigned integer. The format of this optional field (as well as the possible values) is the same as the **compressionFlags** field described in the Fast-Path Update structure specified in [MS-RDPBCGR] section 2.2.9.1.2.1.

**size (2 bytes):** A 16-bit, unsigned integer. The format of this field (as well as the possible values) is the same as the **size** field described in the Fast-Path Update structure specified in [MS-RDPBCGR] section 2.2.9.1.2.1.

**numberOrders (2 bytes):** A 16-bit, unsigned integer. The number of [Drawing Order (section 2.2.2.1.1)](#Section_762f3dd36a76485786a4a3ca951ca3a6) structures contained in the **orderData** field.

**orderData (variable):** A variable-sized array of Drawing Order (section 2.2.2.1.1) structures packed on byte boundaries. Each structure contains a primary, secondary, or alternate secondary drawing order. The **controlFlags** field of the Drawing Order identifies the type of drawing order.

##### Drawing Order (DRAWING\_ORDER)

The DRAWING\_ORDER structure is used to describe and encapsulate a single primary, secondary, or alternate secondary drawing order sent from server to client. All drawing orders conform to this basic structure (see sections [2.2.2.2.1.1.2](#Section_23f766d483434e6b8281071ddccc0272), [2.2.2.2.1.2.1.1](#Section_c54e2667335d4a59a6e27dc9744dbe79), and [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| controlFlags | | | | | | | | orderSpecificData (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**controlFlags (1 byte):** An 8-bit, unsigned integer. A control byte that identifies the class of the drawing order.

If the TS\_STANDARD (0x01) flag is set, the order is a primary drawing order. If both the TS\_STANDARD (0x01) and TS\_SECONDARY (0x02) flags are set, the order is a secondary drawing order. Finally, if only the TS\_SECONDARY (0x02) flag is set, the order is an alternate secondary drawing order.

More flags MAY be present, depending on the drawing order class. The flags listed are common to all three classes of drawing orders.

| Name | Value |
| --- | --- |
| TS\_STANDARD | 0x01 |
| TS\_SECONDARY | 0x02 |

**orderSpecificData (variable):** Variable-length data specific to the drawing order class and the drawing order itself.

###### Primary Drawing Orders

Common Data Types

Coord Field (COORD\_FIELD)

The COORD\_FIELD structure is used to describe a value in the range -32768 to 32767.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| signedValue (variable) | | | | | | | | | | | | | | | |

**signedValue (variable):** A signed, 1-byte or 2-byte value that describes a coordinate in the range -32768 to 32767.

When the **controlFlags** field (see section [2.2.2.2.1.1.2](#Section_23f766d483434e6b8281071ddccc0272)) of the primary drawing order that contains the COORD\_FIELD structure has the TS\_DELTA\_COORDINATES flag (0x10) set, the **signedValue** field MUST contain a signed 1-byte value. If the TS\_DELTA\_COORDINATES flag is not set, the **signedValue** field MUST contain a 2-byte signed value.

The 1-byte format contains a signed delta from the previous value of the Coord field. To obtain the new value of the field, the decoder MUST increment the previous value of the field by the signed delta to produce the current value. The 2-byte format is simply the full value of the field that MUST replace the previous value.

One-Byte Header Variable Field (VARIABLE1\_FIELD)

TheVARIABLE1\_FIELD structure is used to encode a variable-length byte-stream that will hold a maximum of 255 bytes. This structure is always situated at the end of an order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cbData | | | | | | | | rgbData (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**cbData (1 byte):** An 8-bit, unsigned integer. The number of bytes present in the **rgbData** field.

**rgbData (variable):** Variable-length, binary data. The size of this data, in bytes, is given by the **cbData** field.

Two-Byte Header Variable Field (VARIABLE2\_FIELD)

The VARIABLE2\_FIELD structure is used to encode a variable-length byte-stream that holds a maximum of 32,767 bytes. This structure is always situated at the end of an order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cbData | | | | | | | | | | | | | | | | rgbData (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**cbData (2 bytes):** A 16-bit, unsigned integer. The number of bytes present in the **rgbData** field.

**rgbData (variable):** Variable-length, binary data. The size of this data, in bytes, is given by the **cbData** field.

Delta-Encoded Points (DELTA\_PTS\_FIELD)

The DELTA\_PTS\_FIELD structure is used to encode a series of points. Each point is expressed as an X and Y delta from the previous point in the series (the first X and Y deltas are relative to a base point that MUST be included in the order that contains the DELTA\_PTS\_FIELD structure). The number of points is order-dependent and is not specified by any field within the DELTA\_PTS\_FIELD structure. Instead, a separate field within the order that contains the DELTA\_PTS\_FIELD structure MUST be used to specify the number of points (this field SHOULD[<1>](#Appendix_A_1) be placed immediately before the DELTA\_PTS\_FIELD structure in the order encoding).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| zeroBits (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| deltaEncodedPoints (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**zeroBits (variable):** A variable-length byte field. The **zeroBits** field is used to indicate the absence of an X or Y delta value for a specific point in the series. The size in bytes of the **zeroBits** field is given by ceil(NumPoints / 4) where NumPoints is the number of points being encoded. Each point in the series requires two zero-bits (four points per byte) to indicate whether an X or Y delta value is zero (and not present), starting with the most significant bits, so that for the first point the X-zero flag is set at (zeroBits[0] & 0x80), and the Y-zero flag is set at (zeroBits[0] & 0x40).

**deltaEncodedPoints (variable):** A variable-length byte field. The **deltaEncodedPoints** field contains a series of (X, Y) pairs, each pair specifying the delta from the previous pair in the series (the first pair in the series contains a delta from a pre-established coordinate).

The presence of the X and Y delta values for a given pair in the series is dictated by the individual bits of the **zeroBits** field. If the zero bit is set for a given X or Y component, its value is unchanged from the previous X or Y component in the series (a delta of zero), and no data is provided. If the zero bit is not set for a given X or Y component, the delta value it represents is encoded in a packed signed format:

* If the high bit (0x80) is not set in the first encoding byte, the field is 1 byte long and is encoded as a signed delta in the lower 7 bits of the byte.
* If the high bit of the first encoding byte is set, the lower 7 bits of the first byte and the 8 bits of the next byte are concatenated (the first byte containing the high-order bits) to create a 15-bit signed delta value.

Delta-Encoded Rectangles (DELTA\_RECTS\_FIELD)

The DELTA\_RECTS\_FIELD structure is used to encode a series of rectangles. Each rectangle is encoded as a (left, top, width, height) quadruple with the left and top components of each quadruple containing the delta from the left and top components of the previous rectangle; the first rectangle in the series is implicitly assumed to be (0, 0, 0, 0). The number of rectangles is order-dependent and not specified by any field within the DELTA\_RECTS\_FIELD structure. Instead, a separate field within the order that contains the DELTA\_RECTS\_FIELD structure MUST be used to specify the number of rectangles (this field SHOULD be placed immediately before the DELTA\_RECTS\_FIELD structure in the order encoding). The maximum number of rectangles that can be encoded by this structure is 45.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| zeroBits (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| deltaEncodedRectangles (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**zeroBits (variable):** A variable-length byte field. The **zeroBits** field is used to indicate the absence of a left, top, width, or height component. The size, in bytes, of the **zeroBits** field is given by ceil(NumRects / 2) where NumRects is the number of rectangles being encoded. Each rectangle in the series requires four zero-bits (two rectangles per byte) to indicate whether a left, top, width, or height component is zero (and not present), starting with the most significant bits, so that for the first rectangle the left-zero flag is set at (**zeroBits**[0] & 0x80), the top-zero flag is set at (**zeroBits**[0] & 0x40), the width-zero flag is set at (**zeroBits**[0] & 0x20), and the height-zero flag is set at (**zeroBits**[0] & 0x10).

**deltaEncodedRectangles (variable):** A variable-length byte field. The **deltaEncodedRectangles** field contains a series of (left, top, width, height) quadruples with the left and top components in each quadruple specifying the delta from the left and top components of the previous rectangle in the series; the first rectangle in the series is implicitly assumed to be (0, 0, 0, 0).

Assume there are two rectangles specified in (left, top, right, bottom) quadruples:

1. 1: (L1, T1, R1, B1)
2. 2: (L2, T2, R2, B2)

Assuming Rectangle 1 is the first in the series, and by using the (left, top, width, height) quadruple encoding scheme, these two rectangles would be specified as:

1. 1: (L1, T1, R1 - L1, B1 - T1)
2. 2: (L2 - L1, T2 - T1, R2 - L2, B2 - T2)

The presence of the left, top, width, or height component for a given quadruple is dictated by the individual bits of the **zeroBits** field. If the zero bit is set for a given left, top, width, or height component, its value is unchanged from the previous corresponding left, top, width, or height value in the series (a delta of zero), and no data is provided. If the zero bit is not set for a left, right, width, or height component, its value is encoded in a packed signed format:

* If the high bit (0x80) is not set in the first encoding byte, the field is 1 byte long and is encoded as a signed delta in the lower 7 bits of the byte.
* If the high bit of the first encoding byte is set, the lower 7 bits of the first byte and the 8 bits of the next byte are concatenated (the first byte containing the high order bits) to create a 15-bit signed delta value.

Binary Raster Operation (ROP2\_OPERATION)

The ROP2\_OPERATION structure is used to define how the bits in a destination bitmap and a selected brush or pen are combined by using Boolean operators.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| rop2Operation | | | | | | | |

**rop2Operation (1 byte):** An 8-bit, unsigned integer. A raster-operation code that describes a Boolean operation, in [**Reverse Polish Notation**](#gt_0a595f18-03ac-4b23-b9f4-9924a0f1a4e6), to perform on the bits in a destination bitmap (D) and selected brush or pen (P). This operation is a combination of the AND (a), OR (o), NOT (n), and XOR (x) Boolean operators.

| Value | Meaning |
| --- | --- |
| R2\_BLACK  0x01 | 0 |
| R2\_NOTMERGEPEN  0x02 | DPon |
| R2\_MASKNOTPEN  0x03 | DPna |
| R2\_NOTCOPYPEN  0x04 | Pn |
| R2\_MASKPENNOT  0x05 | PDna |
| R2\_NOT  0x06 | Dn |
| R2\_XORPEN  0x07 | DPx |
| R2\_NOTMASKPEN  0x08 | DPan |
| R2\_MASKPEN  0x09 | DPa |
| R2\_NOTXORPEN  0x0A | DPxn |
| R2\_NOP  0x0B | D |
| R2\_MERGENOTPEN  0x0C | DPno |
| R2\_COPYPEN  0x0D | P |
| R2\_MERGEPENNOT  0x0E | PDno |
| R2\_MERGEPEN  0x0F | PDo |
| R2\_WHITE  0x10 | 1 |

For example, by using the previous table, it can be determined that the R2\_MERGEPEN (0x0F) operation replaces the values of the pixels in the destination bitmap with a combination of pixel values of the destination and pen.

For more information about binary raster operations, see the material on binary raster operations in [YUAN] section 8.1.

Ternary Raster Operation Index (ROP3\_OPERATION\_INDEX)

The ROP3\_OPERATION\_INDEX structure is used to define how the bits in a source bitmap, destination bitmap, and a selected brush or pen are combined by using Boolean operators.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| rop3Index | | | | | | | |

**rop3Index (1 byte):** An 8-bit, unsigned integer. This field contains an index of a raster operation code that describes a Boolean operation, in Reverse Polish Notation, to perform on the bits in a source bitmap (S), destination bitmap (D), and selected brush or pen (P). This operation is a combination of the AND (a), OR (o), NOT (n), and XOR (x) Boolean operators.

| Value | Raster-operation Code (ROP) / Boolean Function in Reverse Polish Notation (RPN) |
| --- | --- |
| 0x00 | ROP: 0x00000042 (BLACKNESS)  RPN: 0 |
| 0x01 | ROP: 0x00010289  RPN: DPSoon |
| 0x02 | ROP: 0x00020C89  RPN: DPSona |
| 0x03 | ROP: 0x000300AA  RPN: PSon |
| 0x04 | ROP: 0x00040C88  RPN: SDPona |
| 0x05 | ROP: 0x000500A9  RPN: DPon |
| 0x06 | ROP: 0x00060865  RPN: PDSxnon |
| 0x07 | ROP: 0x000702C5  RPN: PDSaon |
| 0x08 | ROP: 0x00080F08  RPN: SDPnaa |
| 0x09 | ROP: 0x00090245  RPN: PDSxon |
| 0x0A | ROP: 0x000A0329  RPN: DPna |
| 0x0B | ROP: 0x000B0B2A  RPN: PSDnaon |
| 0x0C | ROP: 0x000C0324  RPN: SPna |
| 0x0D | ROP: 0x000D0B25  RPN: PDSnaon |
| 0x0E | ROP: 0x000E08A5  RPN: PDSonon |
| 0x0F | ROP: 0x000F0001  RPN: Pn |
| 0x10 | ROP: 0x00100C85  RPN: PDSona |
| 0x11 | ROP: 0x001100A6 (NOTSRCERASE)  RPN: DSon |
| 0x12 | ROP: 0x00120868  RPN: SDPxnon |
| 0x13 | ROP: 0x001302C8  RPN: SDPaon |
| 0x14 | ROP: 0x00140869  RPN: DPSxnon |
| 0x15 | ROP: 0x001502C9  RPN: DPSaon |
| 0x16 | ROP: 0x00165CCA  RPN: PSDPSanaxx |
| 0x17 | ROP: 0x00171D54  RPN: SSPxDSxaxn |
| 0x18 | ROP: 0x00180D59  RPN: SPxPDxa |
| 0x19 | ROP: 0x00191CC8  RPN: SDPSanaxn |
| 0x1A | ROP: 0x001A06C5  RPN: PDSPaox |
| 0x1B | ROP: 0x001B0768  RPN: SDPSxaxn |
| 0x1C | ROP: 0x001C06CA  RPN: PSDPaox |
| 0x1D | ROP: 0x001D0766  RPN: DSPDxaxn |
| 0x1E | ROP: 0x001E01A5  RPN: PDSox |
| 0x1F | ROP: 0x001F0385  RPN: PDSoan |
| 0x20 | ROP: 0x00200F09  RPN: DPSnaa |
| 0x21 | ROP: 0x00210248  RPN: SDPxon |
| 0x22 | ROP: 0x00220326  RPN: DSna |
| 0x23 | ROP: 0x00230B24  RPN: SPDnaon |
| 0x24 | ROP: 0x00240D55  RPN: SPxDSxa |
| 0x25 | ROP: 0x00251CC5  RPN: PDSPanaxn |
| 0x26 | ROP: 0x002606C8  RPN: SDPSaox |
| 0x27 | ROP: 0x00271868  RPN: SDPSxnox |
| 0x28 | ROP: 0x00280369  RPN: DPSxa |
| 0x29 | ROP: 0x002916CA  RPN: PSDPSaoxxn |
| 0x2A | ROP: 0x002A0CC9  RPN: DPSana |
| 0x2B | ROP: 0x002B1D58  RPN: SSPxPDxaxn |
| 0x2C | ROP: 0x002C0784  RPN: SPDSoax |
| 0x2D | ROP: 0x002D060A  RPN: PSDnox |
| 0x2E | ROP: 0x002E064A  RPN: PSDPxox |
| 0x2F | ROP: 0x002F0E2A  RPN: PSDnoan |
| 0x30 | ROP: 0x0030032A  RPN: PSna |
| 0x31 | ROP: 0x00310B28  RPN: SDPnaon |
| 0x32 | ROP: 0x00320688  RPN: SDPSoox |
| 0x33 | ROP: 0x00330008 (NOTSRCCOPY)  RPN: Sn |
| 0x34 | ROP: 0x003406C4  RPN: SPDSaox |
| 0x35 | ROP: 0x00351864  RPN: SPDSxnox |
| 0x36 | ROP: 0x003601A8  RPN: SDPox |
| 0x37 | ROP: 0x00370388  RPN: SDPoan |
| 0x38 | ROP: 0x0038078A  RPN: PSDPoax |
| 0x39 | ROP: 0x00390604  RPN: SPDnox |
| 0x3A | ROP: 0x003A0644  RPN: SPDSxox |
| 0x3B | ROP: 0x003B0E24  RPN: SPDnoan |
| 0x3C | ROP: 0x003C004A  RPN: PSx |
| 0x3D | ROP: 0x003D18A4  RPN: SPDSonox |
| 0x3E | ROP: 0x003E1B24  RPN: SPDSnaox |
| 0x3F | ROP: 0x003F00EA  RPN: PSan |
| 0x40 | ROP: 0x00400F0A  RPN: PSDnaa |
| 0x41 | ROP: 0x00410249  RPN: DPSxon |
| 0x42 | ROP: 0x00420D5D  RPN: SDxPDxa |
| 0x43 | ROP: 0x00431CC4  RPN: SPDSanaxn |
| 0x44 | ROP: 0x00440328 (SRCERASE)  RPN: SDna |
| 0x45 | ROP: 0x00450B29  RPN: DPSnaon |
| 0x46 | ROP: 0x004606C6  RPN: DSPDaox |
| 0x47 | ROP: 0x0047076A  RPN: PSDPxaxn |
| 0x48 | ROP: 0x00480368  RPN: SDPxa |
| 0x49 | ROP: 0x004916C5  RPN: PDSPDaoxxn |
| 0x4A | ROP: 0x004A0789  RPN: DPSDoax |
| 0x4B | ROP: 0x004B0605  RPN: PDSnox |
| 0x4C | ROP: 0x004C0CC8  RPN: SDPana |
| 0x4D | ROP: 0x004D1954  RPN: SSPxDSxoxn |
| 0x4E | ROP: 0x004E0645  RPN: PDSPxox |
| 0x4F | ROP: 0x004F0E25  RPN: PDSnoan |
| 0x50 | ROP: 0x00500325  RPN: PDna |
| 0x51 | ROP: 0x00510B26  RPN: DSPnaon |
| 0x52 | ROP: 0x005206C9  RPN: DPSDaox |
| 0x53 | ROP: 0x00530764  RPN: SPDSxaxn |
| 0x54 | ROP: 0x005408A9  RPN: DPSonon |
| 0x55 | ROP: 0x00550009 (DSTINVERT)  RPN: Dn |
| 0x56 | ROP: 0x005601A9  RPN: DPSox |
| 0x57 | ROP: 0x00570389  RPN: DPSoan |
| 0x58 | ROP: 0x00580785  RPN: PDSPoax |
| 0x59 | ROP: 0x00590609  RPN: DPSnox |
| 0x5A | ROP: 0x005A0049 (PATINVERT)  RPN: DPx |
| 0x5B | ROP: 0x005B18A9  RPN: DPSDonox |
| 0x5C | ROP: 0x005C0649  RPN: DPSDxox |
| 0x5D | ROP: 0x005D0E29  RPN: DPSnoan |
| 0x5E | ROP: 0x005E1B29  RPN: DPSDnaox |
| 0x5F | ROP: 0x005F00E9  RPN: DPan |
| 0x60 | ROP: 0x00600365  RPN: PDSxa |
| 0x61 | ROP: 0x006116C6  RPN: DSPDSaoxxn |
| 0x62 | ROP: 0x00620786  RPN: DSPDoax |
| 0x63 | ROP: 0x00630608  RPN: SDPnox |
| 0x64 | ROP: 0x00640788  RPN: SDPSoax |
| 0x65 | ROP: 0x00650606  RPN: DSPnox |
| 0x66 | ROP: 0x00660046 (SRCINVERT)  RPN: DSx |
| 0x67 | ROP: 0x006718A8  RPN: SDPSonox |
| 0x68 | ROP: 0x006858A6  RPN: DSPDSonoxxn |
| 0x69 | ROP: 0x00690145  RPN: PDSxxn |
| 0x6A | ROP: 0x006A01E9  RPN: DPSax |
| 0x6B | ROP: 0x006B178A  RPN: PSDPSoaxxn |
| 0x6C | ROP: 0x006C01E8  RPN: SDPax |
| 0x6D | ROP: 0x006D1785  RPN: PDSPDoaxxn |
| 0x6E | ROP: 0x006E1E28  RPN: SDPSnoax |
| 0x6F | ROP: 0x006F0C65  RPN: PDSxnan |
| 0x70 | ROP: 0x00700CC5  RPN: PDSana |
| 0x71 | ROP: 0x00711D5C  RPN: SSDxPDxaxn |
| 0x72 | ROP: 0x00720648  RPN: SDPSxox |
| 0x73 | ROP: 0x00730E28  RPN: SDPnoan |
| 0x74 | ROP: 0x00740646  RPN: DSPDxox |
| 0x75 | ROP: 0x00750E26  RPN: DSPnoan |
| 0x76 | ROP: 0x00761B28  RPN: SDPSnaox |
| 0x77 | ROP: 0x007700E6  RPN: DSan |
| 0x78 | ROP: 0x007801E5  RPN: PDSax |
| 0x79 | ROP: 0x00791786  RPN: DSPDSoaxxn |
| 0x7A | ROP: 0x007A1E29  RPN: DPSDnoax |
| 0x7B | ROP: 0x007B0C68  RPN: SDPxnan |
| 0x7C | ROP: 0x007C1E24  RPN: SPDSnoax |
| 0x7D | ROP: 0x007D0C69  RPN: DPSxnan |
| 0x7E | ROP: 0x007E0955  RPN: SPxDSxo |
| 0x7F | ROP: 0x007F03C9  RPN: DPSaan |
| 0x80 | ROP: 0x008003E9  RPN: DPSaa |
| 0x81 | ROP: 0x00810975  RPN: SPxDSxon |
| 0x82 | ROP: 0x00820C49  RPN: DPSxna |
| 0x83 | ROP: 0x00831E04  RPN: SPDSnoaxn |
| 0x84 | ROP: 0x00840C48  RPN: SDPxna |
| 0x85 | ROP: 0x00851E05  RPN: PDSPnoaxn |
| 0x86 | ROP: 0x008617A6  RPN: DSPDSoaxx |
| 0x87 | ROP: 0x008701C5  RPN: PDSaxn |
| 0x88 | ROP: 0x008800C6 (SRCAND)  RPN: DSa |
| 0x89 | ROP: 0x00891B08  RPN: SDPSnaoxn |
| 0x8A | ROP: 0x008A0E06  RPN: DSPnoa |
| 0x8B | ROP: 0x008B0666  RPN: DSPDxoxn |
| 0x8C | ROP: 0x008C0E08  RPN: SDPnoa |
| 0x8D | ROP: 0x008D0668  RPN: SDPSxoxn |
| 0x8E | ROP: 0x008E1D7C  RPN: SSDxPDxax |
| 0x8F | ROP: 0x008F0CE5  RPN: PDSanan |
| 0x90 | ROP: 0x00900C45  RPN: PDSxna |
| 0x91 | ROP: 0x00911E08  RPN: SDPSnoaxn |
| 0x92 | ROP: 0x009217A9  RPN: DPSDPoaxx |
| 0x93 | ROP: 0x009301C4  RPN: SPDaxn |
| 0x94 | ROP: 0x009417AA  RPN: PSDPSoaxx |
| 0x95 | ROP: 0x009501C9  RPN: DPSaxn |
| 0x96 | ROP: 0x00960169  RPN: DPSxx |
| 0x97 | ROP: 0x0097588A  RPN: PSDPSonoxx |
| 0x98 | ROP: 0x00981888  RPN: SDPSonoxn |
| 0x99 | ROP: 0x00990066  RPN: DSxn |
| 0x9A | ROP: 0x009A0709  RPN: DPSnax |
| 0x9B | ROP: 0x009B07A8  RPN: SDPSoaxn |
| 0x9C | ROP: 0x009C0704  RPN: SPDnax |
| 0x9D | ROP: 0x009D07A6  RPN: DSPDoaxn |
| 0x9E | ROP: 0x009E16E6  RPN: DSPDSaoxx |
| 0x9F | ROP: 0x009F0345  RPN: PDSxan |
| 0xA0 | ROP: 0x00A000C9  RPN: DPa |
| 0xA1 | ROP: 0x00A11B05  RPN: PDSPnaoxn |
| 0xA2 | ROP: 0x00A20E09  RPN: DPSnoa |
| 0xA3 | ROP: 0x00A30669  RPN: DPSDxoxn |
| 0xA4 | ROP: 0x00A41885  RPN: PDSPonoxn |
| 0xA5 | ROP: 0x00A50065  RPN: PDxn |
| 0xA6 | ROP: 0x00A60706  RPN: DSPnax |
| 0xA7 | ROP: 0x00A707A5  RPN: PDSPoaxn |
| 0xA8 | ROP: 0x00A803A9  RPN: DPSoa |
| 0xA9 | ROP: 0x00A90189  RPN: DPSoxn |
| 0xAA | ROP: 0x00AA0029  RPN: D |
| 0xAB | ROP: 0x00AB0889  RPN: DPSono |
| 0xAC | ROP: 0x00AC0744  RPN: SPDSxax |
| 0xAD | ROP: 0x00AD06E9  RPN: DPSDaoxn |
| 0xAE | ROP: 0x00AE0B06  RPN: DSPnao |
| 0xAF | ROP: 0x00AF0229  RPN: DPno |
| 0xB0 | ROP: 0x00B00E05  RPN: PDSnoa |
| 0xB1 | ROP: 0x00B10665  RPN: PDSPxoxn |
| 0xB2 | ROP: 0x00B21974  RPN: SSPxDSxox |
| 0xB3 | ROP: 0x00B30CE8  RPN: SDPanan |
| 0xB4 | ROP: 0x00B4070A  RPN: PSDnax |
| 0xB5 | ROP: 0x00B507A9  RPN: DPSDoaxn |
| 0xB6 | ROP: 0x00B616E9  RPN: DPSDPaoxx |
| 0xB7 | ROP: 0x00B70348  RPN: SDPxan |
| 0xB8 | ROP: 0x00B8074A  RPN: PSDPxax |
| 0xB9 | ROP: 0x00B906E6  RPN: DSPDaoxn |
| 0xBA | ROP: 0x00BA0B09  RPN: DPSnao |
| 0xBB | ROP: 0x00BB0226 (MERGEPAINT)  RPN: DSno |
| 0xBC | ROP: 0x00BC1CE4  RPN: SPDSanax |
| 0xBD | ROP: 0x00BD0D7D  RPN: SDxPDxan |
| 0xBE | ROP: 0x00BE0269  RPN: DPSxo |
| 0xBF | ROP: 0x00BF08C9  RPN: DPSano |
| 0xC0 | ROP: 0x00C000CA (MERGECOPY)  RPN: PSa |
| 0xC1 | ROP: 0x00C11B04  RPN: SPDSnaoxn |
| 0xC2 | ROP: 0x00C21884  RPN: SPDSonoxn |
| 0xC3 | ROP: 0x00C3006A  RPN: PSxn |
| 0xC4 | ROP: 0x00C40E04  RPN: SPDnoa |
| 0xC5 | ROP: 0x00C50664  RPN: SPDSxoxn |
| 0xC6 | ROP: 0x00C60708  RPN: SDPnax |
| 0xC7 | ROP: 0x00C707AA  RPN: PSDPoaxn |
| 0xC8 | ROP: 0x00C803A8  RPN: SDPoa |
| 0xC9 | ROP: 0x00C90184  RPN: SPDoxn |
| 0xCA | ROP: 0x00CA0749  RPN: DPSDxax |
| 0xCB | ROP: 0x00CB06E4  RPN: SPDSaoxn |
| 0xCC | ROP: 0x00CC0020 (SRCCOPY)  RPN: S |
| 0xCD | ROP: 0x00CD0888  RPN: SDPono |
| 0xCE | ROP: 0x00CE0B08  RPN: SDPnao |
| 0xCF | ROP: 0x00CF0224  RPN: SPno |
| 0xD0 | ROP: 0x00D00E0A  RPN: PSDnoa |
| 0xD1 | ROP: 0x00D1066A  RPN: PSDPxoxn |
| 0xD2 | ROP: 0x00D20705  RPN: PDSnax |
| 0xD3 | ROP: 0x00D307A4  RPN: SPDSoaxn |
| 0xD4 | ROP: 0x00D41D78  RPN: SSPxPDxax |
| 0xD5 | ROP: 0x00D50CE9  RPN: DPSanan |
| 0xD6 | ROP: 0x00D616EA  RPN: PSDPSaoxx |
| 0xD7 | ROP: 0x00D70349  RPN: DPSxan |
| 0xD8 | ROP: 0x00D80745  RPN: PDSPxax |
| 0xD9 | ROP: 0x00D906E8  RPN: SDPSaoxn |
| 0xDA | ROP: 0x00DA1CE9  RPN: DPSDanax |
| 0xDB | ROP: 0x00DB0D75  RPN: SPxDSxan |
| 0xDC | ROP: 0x00DC0B04  RPN: SPDnao |
| 0xDD | ROP: 0x00DD0228  RPN: SDno |
| 0xDE | ROP: 0x00DE0268  RPN: SDPxo |
| 0xDF | ROP: 0x00DF08C8  RPN: SDPano |
| 0xE0 | ROP: 0x00E003A5  RPN: PDSoa |
| 0xE1 | ROP: 0x00E10185  RPN: PDSoxn |
| 0xE2 | ROP: 0x00E20746  RPN: DSPDxax |
| 0xE3 | ROP: 0x00E306EA  RPN: PSDPaoxn |
| 0xE4 | ROP: 0x00E40748  RPN: SDPSxax |
| 0xE5 | ROP: 0x00E506E5  RPN: PDSPaoxn |
| 0xE6 | ROP: 0x00E61CE8  RPN: SDPSanax |
| 0xE7 | ROP: 0x00E70D79  RPN: SPxPDxan |
| 0xE8 | ROP: 0x00E81D74  RPN: SSPxDSxax |
| 0xE9 | ROP: 0x00E95CE6  RPN: DSPDSanaxxn |
| 0xEA | ROP: 0x00EA02E9  RPN: DPSao |
| 0xEB | ROP: 0x00EB0849  RPN: DPSxno |
| 0xEC | ROP: 0x00EC02E8  RPN: SDPao |
| 0xED | ROP: 0x00ED0848  RPN: SDPxno |
| 0xEE | ROP: 0x00EE0086 (SRCPAINT)  RPN: DSo |
| 0xEF | ROP: 0x00EF0A08  RPN: SDPnoo |
| 0xF0 | ROP: 0x00F00021 (PATCOPY)  RPN: P |
| 0xF1 | ROP: 0x00F10885  RPN: PDSono |
| 0xF2 | ROP: 0x00F20B05  RPN: PDSnao |
| 0xF3 | ROP: 0x00F3022A  RPN: PSno |
| 0xF4 | ROP: 0x00F40B0A  RPN: PSDnao |
| 0xF5 | ROP: 0x00F50225  RPN: PDno |
| 0xF6 | ROP: 0x00F60265  RPN: PDSxo |
| 0xF7 | ROP: 0x00F708C5  RPN: PDSano |
| 0xF8 | ROP: 0x00F802E5  RPN: PDSao |
| 0xF9 | ROP: 0x00F90845  RPN: PDSxno |
| 0xFA | ROP: 0x00FA0089  RPN: DPo |
| 0xFB | ROP: 0x00FB0A09 (PATPAINT)  RPN: DPSnoo |
| 0xFC | ROP: 0x00FC008A  RPN: PSo |
| 0xFD | ROP: 0x00FD0A0A  RPN: PSDnoo |
| 0xFE | ROP: 0x00FE02A9  RPN: DPSoo |
| 0xFF | ROP: 0x00FF0062 (WHITENESS)  RPN: 1 |

For example, by using the previous table, it can be determined that the raster operation at index 0xEE (SRCPAINT) replaces the values of the pixels in the destination bitmap with a combination of pixel values of the destination and source bitmaps.

For more information about ternary raster operations, see the material on ternary raster operations in [YUAN] section 11.1.

Generic Color (TS\_COLOR)

The TS\_COLOR structure holds a 3-byte RGB color triplet (the red, green, and blue components necessary to reproduce a color in the additive RGB space) or a 1-byte palette index.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| RedOrPaletteIndex | | | | | | | | Green | | | | | | | | Blue | | | | | | | |

**RedOrPaletteIndex (1 byte):** An 8-bit, unsigned integer. **RedOrPaletteIndex** is used as a palette index for 16-color and 256-color palettized color schemes. If the RGB color scheme is in effect, this field contains the red RGB component. To determine whether a palettized or RGB color scheme is in effect, the client MUST examine the **preferredBitsPerPixel** field of the Bitmap Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.2). If **preferredBitsPerPixel** is less than or equal to 8, then a palettized color scheme is in effect; otherwise, an RGB color scheme is in effect.

**Green (1 byte):** An 8-bit, unsigned integer. **Green** contains the green RGB color component.

**Blue (1 byte):** An 8-bit, unsigned integer. **Blue** contains the blue RGB color component.

Fill Mode (FILL\_MODE)

The FILL\_MODE structure specifies the algorithm used to determine if a given point is contained within a polygon.

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

**FillMode (1 byte):** An 8-bit, unsigned integer that specifies the fill mode. **FillMode** MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| ALTERNATE  0x01 | Alternate fill mode. To test if a point is inside a polygon, a ray is drawn from that point parallel to the x-axis to infinity, and the number of intersections between the ray and the polygon outline is counted. If the number of intersections is odd, the point is inside the polygon, and, if it is even, the point is outside. |
| WINDING  0x02 | Winding fill mode. To determine if a point is within a polygon, a ray is drawn from that point parallel to the x-axis to infinity, and its intersections with the outline are examined. A count that starts from 0 is kept for the ray. Each intersection with a clockwise polygon outline increments the count, and each intersection with a counterclockwise polygon outline decrements the count. If the final value of the count is nonzero, the point is inside the polygon; otherwise, it is outside. |

For alternate and winding fill modes, see [YUAN] section 9.4 and [PETZOLD] pages 169–171.

Primary Drawing Order (PRIMARY\_DRAWING\_ORDER)

The PRIMARY\_DRAWING\_ORDER structure encapsulates a primary drawing order. All primary drawing orders employ special field encoding to reduce the number of bytes sent on the wire. Field encoding maintains a copy of the most recent field values that were used in each primary drawing order, in addition to common state information such as the last bounding rectangle used across all orders and the last order type. Only the fields that have changed are sent on the wire. This implies that all the fields in a primary drawing order are optional, their presence being denoted by the **controlFlags** and **fieldFlags** fields.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| controlFlags | | | | | | | | orderType (optional) | | | | | | | | fieldFlags (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bounds (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| primaryOrderData (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**controlFlags (1 byte):** An 8-bit, unsigned integer. A control byte that identifies the class of the drawing order and describes the fields that are included in the order and the type of coordinates being used. This field MUST contain a combination of the following flags.

| Value | Meaning |
| --- | --- |
| TS\_STANDARD  0x01 | Indicates that the order is a primary drawing order. This flag MUST be set. |
| TS\_BOUNDS  0x04 | Indicates that the order has a bounding rectangle. |
| TS\_TYPE\_CHANGE  0x08 | Indicates that the order type has changed and that the **orderType** field is present. |
| TS\_DELTA\_COORDINATES  0x10 | Indicates that all of the Coord-type fields in the order (see section [2.2.2.2.1.1.1.1](#Section_870b982c9abe40cb99d02c3f0cc5fb74)) are specified as 1-byte signed deltas from their previous values. |
| TS\_ZERO\_BOUNDS\_DELTAS  0x20 | Indicates that the previous bounding rectangle MUST be used, as the bounds have not changed (this implies that the bounds field is not present). This flag is only applicable if the TS\_BOUNDS (0x04) flag is set. |
| TS\_ZERO\_FIELD\_BYTE\_BIT0  0x40 | Used in conjunction with the TS\_ZERO\_FIELD\_BYTE\_BIT1 (0x80) flag to form a 2-bit count (so maximum of 3) of the number of field flag bytes (present in the **fieldFlags** field) that are zero and not present, counted from the end of the set of field flag bytes. This flag is the least significant bit of the count. |
| TS\_ZERO\_FIELD\_BYTE\_BIT1  0x80 | Used in conjunction with the TS\_ZERO\_FIELD\_BYTE\_BIT0 (0x40) flag to form a 2-bit count (so maximum of 3) of the number of field flag bytes (present in the **fieldFlags** field) that are zero and not present, counted from the end of the set of field flag bytes. This flag is the most significant bit of the count. |

For the use of the TS\_ZERO\_FIELD\_BYTE\_BIT0 (0x40) and TS\_ZERO\_FIELD\_BYTE\_BIT1 (0x80) flags, see the **fieldFlags** field.

**orderType (1 byte):** An 8-bit, unsigned integer. An optional identifier describing the type of primary drawing order. The initial value for the **orderType** agreed on by both the server and client is TS\_ENC\_PATBLT\_ORDER (0x01) for the PatBlt primary drawing order.

| Value | Meaning |
| --- | --- |
| TS\_ENC\_DSTBLT\_ORDER  0x00 | [DstBlt (section 2.2.2.2.1.1.2.1)](#Section_87ea30df59d6438ea73583f0225fbf91) Primary Drawing Order. |
| TS\_ENC\_PATBLT\_ORDER  0x01 | [PatBlt (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8) Primary Drawing Order. |
| TS\_ENC\_SCRBLT\_ORDER  0x02 | [ScrBlt (section 2.2.2.2.1.1.2.7)](#Section_a4e322b0cd644dfc8e1af24dc0edc99d) Primary Drawing Order. |
| TS\_ENC\_DRAWNINEGRID\_ORDER  0x07 | [DrawNineGrid (section 2.2.2.2.1.1.2.21)](#Section_44bd6a68a5b74a38980706ba4ab4a72a) Primary Drawing Order. |
| TS\_ENC\_MULTI\_DRAWNINEGRID\_ORDER  0x08 | [MultiDrawNineGrid (section 2.2.2.2.1.1.2.22)](#Section_101fecdc3ad841c6b7b670d8efd4ad70) Primary Drawing Order. |
| TS\_ENC\_LINETO\_ORDER  0x09 | [LineTo (section 2.2.2.2.1.1.2.11)](#Section_27276fcf32be49a99a1797ec6167ac7e) Primary Drawing Order. |
| TS\_ENC\_OPAQUERECT\_ORDER  0x0A | [OpaqueRect (section 2.2.2.2.1.1.2.5)](#Section_1eead7aaac63411a9f8cb1b227526877) Primary Drawing Order. |
| TS\_ENC\_SAVEBITMAP\_ORDER  0x0B | [SaveBitmap (section 2.2.2.2.1.1.2.12)](#Section_db4ce8dbcf76483d99ba354cd010ea4a) Primary Drawing Order. |
| TS\_ENC\_MEMBLT\_ORDER  0x0D | [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) Primary Drawing Order. |
| TS\_ENC\_MEM3BLT\_ORDER  0x0E | [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Order. |
| TS\_ENC\_MULTIDSTBLT\_ORDER  0x0F | [MultiDstBlt (section 2.2.2.2.1.1.2.2)](#Section_aaae78403eb440389204b95f87b44534) Primary Drawing Order. |
| TS\_ENC\_MULTIPATBLT\_ORDER  0x10 | [MultiPatBlt (section 2.2.2.2.1.1.2.4)](#Section_5f677576a62e48d9a58d711490882365) Primary Drawing Order. |
| TS\_ENC\_MULTISCRBLT\_ORDER  0x11 | [MultiScrBlt (section 2.2.2.2.1.1.2.8)](#Section_31b79b4915b74709bbc2d58ce0362292) Primary Drawing Order. |
| TS\_ENC\_MULTIOPAQUERECT\_ORDER  0x12 | [MultiOpaqueRect (section 2.2.2.2.1.1.2.6)](#Section_74e46cd9d6494bbe9f8af4200f23302e) Primary Drawing Order. |
| TS\_ENC\_FAST\_INDEX\_ORDER  0x13 | [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7) Primary Drawing Order. |
| TS\_ENC\_POLYGON\_SC\_ORDER  0x14 | [PolygonSC (section 2.2.2.2.1.1.2.16)](#Section_22e1117fd40d41df970959389ac1f8dc) Primary Drawing Order. |
| TS\_ENC\_POLYGON\_CB\_ORDER  0x15 | [PolygonCB (section 2.2.2.2.1.1.2.17)](#Section_24f4e2291f9f4633b9be66d7579ba36d) Primary Drawing Order. |
| TS\_ENC\_POLYLINE\_ORDER  0x16 | [Polyline (section 2.2.2.2.1.1.2.18)](#Section_b60b4d9baac54b9884d3edc0c0779a5c) Primary Drawing Order. |
| TS\_ENC\_FAST\_GLYPH\_ORDER  0x18 | [FastGlyph (section 2.2.2.2.1.1.2.15)](#Section_839e963ce24a42adb4bdeb108340629c) Primary Drawing Order. |
| TS\_ENC\_ELLIPSE\_SC\_ORDER  0x19 | [EllipseSC (section 2.2.2.2.1.1.2.19)](#Section_5146fab3c43f44d7b54104666a80e33b) Primary Drawing Order. |
| TS\_ENC\_ELLIPSE\_CB\_ORDER  0x1A | [EllipseCB (section 2.2.2.2.1.1.2.20)](#Section_897203838394423a810aba631d66055f) Primary Drawing Order. |
| TS\_ENC\_INDEX\_ORDER  0x1B | [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) Primary Drawing Order. |

**fieldFlags (variable):** A variable-length, 1-byte to 3-byte field. The optional **fieldFlags** field is used to indicate the presence of an order field in the encoded fields portion of the packet (represented by the **primaryOrderData** field). Each bit in the **fieldFlags** field functions as a flag and indicates if a particular order field is present.

The number of bytes used to represent the field flags will vary based on the maximum number of fields that a primary drawing order can hold. For any given drawing order, the size (in bytes) of the **fieldFlags** field MUST be computed with the following formula.

1. ceil(((numberOfOrderFields) + 1) / 8)

The maximum number of allowed fields is 23, which corresponds to three field flags bytes. The "+ 1" in the equation indicates that for seven fields there is one field flags byte; however, for eight fields, there are 2 bytes. (This drives some of the specialized orders in RDP where a special case may be used to reduce the number of fields below 7 or 15, thereby reducing the number of field flags bytes sent.)

The individual field flags bytes are arranged in little-endian order; the low order byte is first. The order of the flags proceeds from 0x01 (the least significant bit), which corresponds to the first field in the order, 0x02 the second field in the order, 0x04 the third field in the order, and so on.

The presence of the **fieldFlags** field is also governed by the **controlFlags** field ZERO\_FIELD\_BYTE\_BIT0 (0x40) and ZERO\_FIELD\_BYTE\_BIT1 (0x80) flags (known as the "zero flags"). These flags are used as a 2-bit count of the number of field flags bytes that are zero and not present, counted from the end of the set of flag bytes.

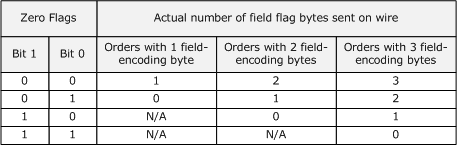


Figure : Relationship between zero flags and the number of flag field bytes sent on the wire

For example, assume that a given order has nine fields. Hence, there can be up to two bytes in the fieldFlags field. Assume further that the fieldFlags field has the field-encoding bytes {0x20, 0x00}. In this case, there is 1 zero byte (counting from the end backward); thus, the **controlFlags** field contains the ZERO\_FIELD\_BYTE\_BIT0 flag (0x40), and the order is sent on the wire with a **fieldFlags** value of 0x20.

**bounds (variable)**: A variable-length, 1-byte to 9-byte field. The presence of the optional **bounds** field is governed by the TS\_BOUNDS (0x04) flag in the **controlFlags** field, which indicates that the order MUST have a bounding region applied. If the **controlFlags** field TS\_ZERO\_BOUNDS\_DELTAS (0x20) flag is also set, the previous bounding rectangle MUST be used, as the bounds have not changed (this implies that the bounds field is not present). Otherwise, the bounds are encoded as an encoding description byte followed by one or more encoded bounds (written in the order: left, top, right, bottom). The description byte MUST contain a TS\_BOUND\_XXX or TS\_BOUND\_DELTA\_XXX flag to describe each of the encoded bounds that are present.

| Flag | Meaning |
| --- | --- |
| TS\_BOUND\_LEFT  0x01 | Indicates that the left bound is present and encoded as a 2-byte, little-endian ordered value. |
| TS\_BOUND\_TOP  0x02 | Indicates that the top bound is present and encoded as a 2-byte, little-endian ordered value. |
| TS\_BOUND\_RIGHT  0x04 | Indicates that the right bound is present and encoded as a 2-byte, little-endian ordered value. |
| TS\_BOUND\_BOTTOM  0x08 | Indicates that the bottom bound is present and encoded as a 2-byte, little-endian ordered value. |
| TS\_BOUND\_DELTA\_LEFT  0x10 | Indicates that the left bound is present and encoded as a 1-byte signed value used as an offset (-128 to 127) from the previous value. |
| TS\_BOUND\_DELTA\_TOP  0x20 | Indicates that the top bound is present and encoded as a 1-byte signed value used as an offset (-128 to 127) from the previous value. |
| TS\_BOUND\_DELTA\_RIGHT  0x40 | Indicates that the right bound is present and encoded as a 1-byte, signed value used as an offset (-128 to 127) from the previous value. |
| TS\_BOUND\_DELTA\_BOTTOM  0x80 | Indicates that the bottom bound is present and encoded as a 1-byte, signed value used as an offset (-128 to 127) from the previous value. |

If for a given component a TS\_BOUND\_XXX or TS\_BOUND\_DELTA\_XXX flag is not present, the component value is the same as the last one used, and no value is included in the encoded bounds. If both the TS\_BOUND\_XXX and TS\_BOUND\_DELTA\_XXX flags are present, the TS\_BOUND\_XXX flag is ignored. Hence, to avoid parsing errors, only one flag MUST be used to describe the format of a given encoded bound.

The initial value for the bounds agreed on by both the server and client is the following zero rectangle.

1. (left, top, right, bottom) = (0, 0, 0, 0)

**primaryOrderData (variable):** A variable-length byte field. The **primaryOrderData** field contains the encoded order field values whose presence is governed by the **fieldFlags** field. The encoded fields, if present, MUST be encoded in the same order as the order description (see sections 2.2.2.2.1.1.2.1 through 2.2.2.2.1.1.2.22). If a field is not present, its value is the same as the last value sent for that order. The initial values for all fields are zero.

DstBlt (DSTBLT\_ORDER)

The DstBlt Primary Drawing Order is used to paint a rectangle by using a destination-only raster operation.

1. Encoding order number: 0 (0x00)
2. Negotiation order number: 0 (0x00)
3. Number of fields: 5
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 9 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| bRop (optional) | | | | | | | |

**nLeftRect (variable):** Left coordinate of the destination rectangle specified using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** Top coordinate of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** Width of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** Height of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** Index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)). The resultant ROP3 operation MUST only depend on the destination bits (there MUST NOT be any dependence on source or pattern bits).

MultiDstBlt (MULTI\_DSTBLT\_ORDER)

The MultiDstBlt Primary Drawing Order is used to paint multiple rectangles by using a destination-only raster operation.

1. Encoding order number: 15 (0x0F)
2. Negotiation order number: 15 (0x0F)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 395 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| bRop (optional) | | | | | | | | nDeltaEntries (optional) | | | | | | | | CodedDeltaList (variable) | | | | | | | | | | | | | | | |

**nLeftRect (variable):** A signed, 1-byte or 2-byte field. Left coordinate of the destination rectangle specified using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** A signed, 1-byte or 2-byte field. Top coordinate of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** A signed, 1-byte or 2-byte field. Width of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** A signed, 1-byte or 2-byte field. Height of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** Index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)). The resultant ROP3 operation MUST only depend on the destination bits (there MUST NOT be any dependence on source or pattern bits).

**nDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of bounding rectangles described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [Two-Byte Header Variable Field (section 2.2.2.2.1.1.1.3)](#Section_aead742ea85847f1a7790311660f84af) structure that encapsulates a [Delta-Encoded Rectangles (section 2.2.2.2.1.1.1.5)](#Section_b89f2058b1804da09bd1aa694c87768c) structure that contains bounding rectangles to use when rendering the order. The number of rectangles described by the Delta-Encoded Rectangles structure is specified by the **nDeltaEntries** field.

PatBlt (PATBLT\_ORDER)

The PatBlt Primary Drawing Order is used to paint a rectangle by using a specified brush and three-way raster operation.

1. Encoding order number: 1 (0x01)
2. Negotiation order number: 1 (0x01)
3. Number of fields: 12
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 26 bytes

Note that the negotiation order number for the PatBlt Primary Drawing Order (0x01) is the same as that for the [OpaqueRect Primary Drawing Order (section 2.2.2.2.1.1.2.5)](#Section_1eead7aaac63411a9f8cb1b227526877). Hence support for PatBlt implies support for OpaqueRect. The converse is also true.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| bRop (optional) | | | | | | | | BackColor (optional) | | | | | | | | | | | | | | | | | | | | | | | |
| ForeColor (optional) | | | | | | | | | | | | | | | | | | | | | | | | BrushOrgX (optional) | | | | | | | |
| BrushOrgY (optional) | | | | | | | | BrushStyle (optional) | | | | | | | | BrushHatch (optional) | | | | | | | | BrushExtra | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**nLeftRect (variable):** Left coordinate of the destination rectangle specified using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** Top coordinate of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** Width of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** Height of the destination rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** Index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)).

**BackColor (3 bytes):** Background color described using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** Foreground color described using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BrushOrgX (1 byte):** An 8-bit, signed integer. The x-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushOrgY (1 byte):** An 8-bit, signed integer. The y-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushStyle (1 byte):** An 8-bit, unsigned integer. The style of the brush used in the drawing operation.

If the TS\_CACHED\_BRUSH (0x80) flag is set in the most significant bit of the BrushStyle field, a brush that was previously cached using the Cache Bitmap Secondary Order (see section [2.2.2.2.1.2.7](#Section_9e7027c4d9894cb09674e2e36c95ebdd)) MUST be used. In this case, the BrushHatch field MUST contain the index of the Brush Cache entry that holds the selected brush to use, and the low nibble of the BrushStyle field MUST contain an identifier describing the color depth of the cached brush.

| Value | Meaning |
| --- | --- |
| BMF\_1BPP  0x1 | 1 bit per pixel |
| BMF\_8BPP  0x3 | 8 bits per pixel |
| BMF\_16BPP  0x4 | 15 or 16 bits per pixel |
| BMF\_24BPP  0x5 | 24 bits per pixel |
| BMF\_32BPP  0x6 | 32 bits per pixel |

If the TS\_CACHED\_BRUSH (0x80) flag is not set in the most significant bit of the BrushStyle field, an 8-by-8-pixel brush MUST be used, and one of the following style identifiers MUST be present in the field.

| Value | Meaning |
| --- | --- |
| BS\_SOLID  0x00 | Solid color brush. The BrushHatch field SHOULD be set to 0. |
| BS\_NULL  0x01 | Hollow brush. The BrushHatch field SHOULD be set to 0. |
| BS\_HATCHED  0x02 | Hatched brush. The hatch pattern MUST be described by the BrushHatch field (there are six possible hatch patterns). |
| BS\_PATTERN  0x03 | Pattern brush. The pixel pattern MUST be described by the BrushExtra and BrushHatch fields. |

**BrushHatch (1 byte):** An 8-bit, unsigned integer. Holds a brush hatch identifier or a Brush Cache index, depending on the contents of the BrushStyle field.

If the TS\_CACHED\_BRUSH (0x80) flag is set in the most significant bit of the BrushStyle field, the BrushHatch value MUST contain the index of the Brush Cache entry that holds the selected brush to use.

If the BrushStyle field is set to BS\_SOLID (0x00) or BS\_NULL (0x01), the BrushHatch field SHOULD be set to 0.

If the BrushStyle field is set to BS\_HATCHED (0x02), the BrushHatch field MUST contain one of the following hatch constants.

| Value | Meaning |
| --- | --- |
| HS\_HORIZONTAL  0x00 | Horizontal hatching as in the following figure.  MS-RDPEGDI_pictab16d5aa-64d3-8ba5-7a81-d787a2345540.png  Figure : Horizontal hatch |
| HS\_VERTICAL  0x01 | Vertical hatching as in the following figure.  MS-RDPEGDI_picte34a7e61-713b-6c78-e3ea-e838f8fa33dc.png  Figure : Vertical hatch |
| HS\_FDIAGONAL  0x02 | Diagonal hatching as in the following figure.  MS-RDPEGDI_pictdae1d3dd-1ce1-5a6b-fef4-66de28ad62ae.png  Figure : 45-degree downward, left-to-right hatch |
| HS\_BDIAGONAL  0x03 | Diagonal hatching as in the following figure.  MS-RDPEGDI_pict7f54da0b-d818-c54e-936e-cc6dd6c56e2e.png  Figure : 45-degree upward, left-to-right hatch |
| HS\_CROSS  0x04 | Cross-hatching as in the following figure.  MS-RDPEGDI_pict61428310-5e27-709a-b744-ad817f2b5cae.png  Figure : Horizontal and vertical cross-hatch |
| HS\_DIAGCROSS  0x05 | Cross-hatching as in the following figure.  MS-RDPEGDI_pictf6f351e5-a92e-1ebc-d1c8-9d2f8cb0f45d.png  Figure : 45-degree cross-hatch |

If the **BrushStyle** field is set to BS\_PATTERN (0x03), the **BrushHatch** field MUST encode the pixel pattern present in the bottom row of the 8x8 pattern brush (the pixel patterns in the top seven rows MUST be encoded in the BrushExtra field). For example, if the bottom row of the pattern brush contains an alternating series of black and white pixels, **BrushHatch** will contain either 0xAA or 0x55.

**BrushExtra (7 bytes):** A byte array of length 7. **BrushExtra** contains an array of bitmap bits that encodes the pixel pattern present in the top seven rows of the 8x8 pattern brush. The pixel pattern present in the bottom row is encoded in the **BrushHatch** field. The **BrushExtra** field is only present if the **BrushStyle** is set to BS\_PATTERN (0x03). The rows are encoded in reverse order, that is, the pixels in the penultimate row are encoded in the first byte, and the pixels in the top row are encoded in the seventh byte. For example, a 45-degree downward sloping left-to-right line would be encoded in **BrushExtra** as { 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 } with **BrushHatch** containing the value 0x01 (the bottom row).

MultiPatBlt (MULTI\_PATBLT\_ORDER)

The MultiPatBlt Primary Drawing Order is used to paint multiple rectangles by using a specified brush and three-way raster operation.

1. Encoding order number: 16 (0x10)
2. Negotiation order number: 16 (0x10)
3. Number of fields: 14
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 412 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| bRop (optional) | | | | | | | | BackColor (optional) | | | | | | | | | | | | | | | | | | | | | | | |
| ForeColor (optional) | | | | | | | | | | | | | | | | | | | | | | | | BrushOrgX (optional) | | | | | | | |
| BrushOrgY (optional) | | | | | | | | BrushStyle (optional) | | | | | | | | BrushHatch (optional) | | | | | | | | BrushExtra (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | nDeltaEntries (optional) | | | | | | | | CodedDeltaList (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**nLeftRect (variable):** The left coordinate of the destination rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** The width of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** The height of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** The index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)).

**BackColor (3 bytes):** The background color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** The foreground color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BrushOrgX (1 byte):** An 8-bit, signed integer. The x-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushOrgY (1 byte):** An 8-bit, signed integer. The y-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushStyle (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushStyle** field of the [PatBlt (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8) Primary Drawing Order.

**BrushHatch (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushHatch** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**BrushExtra (7 bytes):** The contents and format of this field are the same as the **BrushExtra** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**nDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of bounding rectangles described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [Two-Byte Header Variable Field (section 2.2.2.2.1.1.1.3)](#Section_aead742ea85847f1a7790311660f84af) structure that encapsulates a [Delta-Encoded Rectangles (section 2.2.2.2.1.1.1.5)](#Section_b89f2058b1804da09bd1aa694c87768c) structure that contains bounding rectangles to use when rendering the order. The number of rectangles described by the Delta-Encoded Rectangles structure is specified by the **nDeltaEntries** field.

OpaqueRect (OPAQUERECT\_ORDER)

The OpaqueRect Primary Drawing Order is used to paint a rectangle by using an opaque brush.

1. Encoding order number: 10 (0x0A)
2. Negotiation order number: 1 (0x01)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 11 bytes

Note that the negotiation order number for the OpaqueRect Primary Drawing Order (0x01) is the same as that for the [PatBlt Primary Drawing Order (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8). Hence support for OpaqueRect implies support for PatBlt. The converse is also true.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| RedOrPaletteIndex (optional) | | | | | | | | Green (optional) | | | | | | | | Blue (optional) | | | | | | | |

**nLeftRect (variable):** The left coordinate of the destination rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** The width of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** The height of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**RedOrPaletteIndex (1 byte):** An 8-bit, unsigned integer. Used as a palette index for 16-color and 256-color palettized color schemes. If the RGB color scheme is in effect, this field holds the red RGB component.

**Green (1 byte):** An 8-bit, unsigned integer. The green RGB color component.

**Blue (1 byte):** An 8-bit, unsigned integer. The blue RGB color component.

MultiOpaqueRect (MULTI\_OPAQUERECT\_ORDER)

The MultiOpaqueRect Primary Drawing Order is used to paint multiple rectangles by using an opaque brush.

1. Encoding order number: 18 (0x12)
2. Negotiation order number: 18 (0x12)
3. Number of fields: 9
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 397 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| RedOrPaletteIndex (optional) | | | | | | | | Green (optional) | | | | | | | | Blue (optional) | | | | | | | | nDeltaEntries (optional) | | | | | | | |
| CodedDeltaList (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**nLeftRect (variable):** The left coordinate of the destination rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** The width of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** The height of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**RedOrPaletteIndex (1 byte):** An 8-bit, unsigned integer. Used as a palette index for 16-color and 256-color palettized color schemes. If the RGB color scheme is in effect, it holds the red RGB component.

**Green (1 byte):** An 8-bit, unsigned integer. The green RGB color component.

**Blue (1 byte):** An 8-bit, unsigned integer. The blue RGB color component.

**nDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of bounding rectangles described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [Two-Byte Header Variable Field (section 2.2.2.2.1.1.1.3)](#Section_aead742ea85847f1a7790311660f84af) structure that encapsulates a [Delta-Encoded Rectangles (section 2.2.2.2.1.1.1.5)](#Section_b89f2058b1804da09bd1aa694c87768c) structure that contains bounding rectangles to use when rendering the order. The number of rectangles described by the Delta-Encoded Rectangles structure is specified by the **nDeltaEntries** field.

ScrBlt (SCRBLT\_ORDER)

The ScrBlt Primary Drawing Order is used to perform a bit-block transfer from a source region to a destination region. The source surface is always the primary drawing surface, while the target surface is the current target surface—specified by the [Switch Surface Alternate Secondary Drawing Order (section 2.2.2.2.1.3.3)](#Section_ada4d8a9421e48c0a0ed7384fa3cf061).

1. Encoding order number: 2 (0x02)
2. Negotiation order number: 2 (0x02)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 13 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| bRop (optional) | | | | | | | | nXSrc (variable) | | | | | | | | | | | | | | | | nYSrc (variable) | | | | | | | |
| ... | | | | | | | |

**nLeftRect (variable):** The left coordinate of the destination rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** The width of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** The height of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** The index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)). The resultant ROP3 operation MUST only depend on the destination and source bits. (There MUST NOT be any dependence on pattern bits.)

**nXSrc (variable):** The x-coordinate of the source rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nYSrc (variable):** The y-coordinate of the source rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

MultiScrBlt (MULTI\_SCRBLT\_ORDER)

The MultiScrBlt Primary Drawing Order is used to perform multiple bit-block transfers from source regions to destination regions of the screen.

1. Encoding order number: 17 (0x11)
2. Negotiation order number: 17 (0x11)
3. Number of fields: 9
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 399 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nWidth (variable) | | | | | | | | | | | | | | | | nHeight (variable) | | | | | | | | | | | | | | | |
| bRop (optional) | | | | | | | | nXSrc (variable) | | | | | | | | | | | | | | | | nYSrc (variable) | | | | | | | |
| ... | | | | | | | | nDeltaEntries (optional) | | | | | | | | CodedDeltaList (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**nLeftRect (variable):** The left coordinate of the destination rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** The width of the destination rectangle specified by using a Coord Field (see section 2.2.2.2.1.1.1.1).

**nHeight (variable):** The height of the destination rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** The index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)). The resultant ROP3 operation MUST only depend on the destination and source bits. (There MUST NOT be any dependence on pattern bits.)

**nXSrc (variable):** The x-coordinate of the source rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nYSrc (variable):** The y-coordinate of the source rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of bounding rectangles described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [Two-Byte Header Variable Field (section 2.2.2.2.1.1.1.3)](#Section_aead742ea85847f1a7790311660f84af) structure that encapsulates a [Delta-Encoded Rectangles (section 2.2.2.2.1.1.1.5)](#Section_b89f2058b1804da09bd1aa694c87768c) structure that contains bounding rectangles to use when rendering the order. The number of rectangles described by the Delta-Encoded Rectangles structure is specified by the **nDeltaEntries** field.

MemBlt (MEMBLT\_ORDER)

The MemBlt Primary Drawing Order is used to render a bitmap stored in the bitmap cache or offscreen bitmap cache to the screen.

1. Encoding order number: 13 (0x0D)
2. Negotiation order number: 3 (0x03)
3. Number of fields: 9
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 17 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId (optional) | | | | | | | | | | | | | | | | nLeftRect (variable) | | | | | | | | | | | | | | | |
| nTopRect (variable) | | | | | | | | | | | | | | | | nWidth (variable) | | | | | | | | | | | | | | | |
| nHeight (variable) | | | | | | | | | | | | | | | | bRop (optional) | | | | | | | | nXSrc (variable) | | | | | | | |
| ... | | | | | | | | nYSrc (variable) | | | | | | | | | | | | | | | | cacheIndex (optional) | | | | | | | |
| ... | | | | | | | |

**cacheId (2 bytes):** A 16-bit, unsigned integer. The **cacheId** field contains the encoded bitmap cache ID and Color Table Cache entry.

The high byte contains the index of the color table entry to use (cached previously with a [Cache Color Table (section 2.2.2.2.1.2.4)](#Section_49fad30181714f38a68a7394baeb7b15) Secondary Drawing Order), and the low byte contains the ID of the bitmap cache in which the source bitmap is stored (cached previously with a Cache Bitmap - Revision 1 (section [2.2.2.2.1.2.2)](#Section_c9365b320be547c9af1afeba2ea1ee9f) or Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3)](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40) Secondary Drawing Order).

The color table entry MUST be in the range 0 to 5 (inclusive).

If the client advertised support for the Revision 1 bitmap caches (section [3.1.1.1.1](#Section_2BF9258842BD45278B3EB90C56E292D2)) by sending the server the Revision 1 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.1), then the cache ID MUST be in the range 0 to 2 (inclusive). However, if the client advertised support for the Revision 2 bitmap caches (section 3.1.1.1.1) by sending the server the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2), then the cache ID MUST be greater than or equal to 0 and less than the number of bitmap caches being used for the connection (specified by the **NumCellCaches** field of the Revision 2 Bitmap Cache Capability Set).

If the bitmap cache ID is set to TS\_BITMAPCACHE\_SCREEN\_ID (0xFF), the **cacheIndex** field MUST contain the index of an entry in the Offscreen Bitmap Cache that contains the source bitmap.

**nLeftRect (variable):** Left coordinate of the blit rectangle specified using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** Top coordinate of the blit rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** Width of the blit rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** Height of the blit rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** Index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)). The resultant ROP3 operation MUST only depend on the destination and source bits (there MUST NOT be any dependence on pattern bits).

**nXSrc (variable):** The x-coordinate of the source rectangle within the source bitmap specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nYSrc (variable):** The inverted y-coordinate of the source rectangle within the source bitmap specified by using a Coord Field (section 2.2.2.2.1.1.1.1). The actual value of the y-coordinate MUST be computed using the following formula.

1. ActualYSrc = (SourceBitmapHeight - nHeight) - nYSrc

**cacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the source bitmap in the bitmap cache specified by the **cacheId** field.

If this field is set to BITMAPCACHE\_WAITING\_LIST\_INDEX (32767), the last bitmap cache entry MUST contain the bitmap data. Otherwise, this field MUST contain a value that is greater than or equal to 0 and less than the maximum number of entries allowed in the source bitmap cache. If the client advertised support for the Revision 1 bitmap caches (section 3.1.1.1.1) by sending the server the Revision 1 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.1), then the maximum number of entries allowed in each individual Revision 1 bitmap cache is specified in the Revision 1 Bitmap Cache Capability Set by the **Cache0Entries**, **Cache1Entries**, and **Cache2Entries** fields. If the client advertised support for the Revision 2 bitmap caches (section 3.1.1.1.1) by sending the server the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2), then the maximum number of entries allowed in each individual Revision 2 bitmap cache is specified in the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2) by the **BitmapCache0CellInfo**, **BitmapCache1CellInfo**, **BitmapCache2CellInfo**, **BitmapCache3CellInfo**, and **BitmapCache4CellInfo** fields.

If the bitmap cache ID (specified in the **cacheId** field) is set to TS\_BITMAPCACHE\_SCREEN\_ID (0xFF), this field MUST contain the index of an entry in the Offscreen Bitmap Cache that contains the source bitmap.

Mem3Blt (MEM3BLT\_ORDER)

The Mem3Blt Primary Drawing Order is used to render a bitmap stored in the bitmap cache or offscreen bitmap cache to the screen by using a specified brush and three-way raster operation.

1. Encoding order number: 14 (0x0E)
2. Negotiation order number: 4 (0x04)
3. Number of fields: 16
4. Number of field encoding bytes: 3
5. Maximum encoded field length: 34 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId (optional) | | | | | | | | | | | | | | | | nLeftRect (variable) | | | | | | | | | | | | | | | |
| nTopRect (variable) | | | | | | | | | | | | | | | | nWidth (variable) | | | | | | | | | | | | | | | |
| nHeight (variable) | | | | | | | | | | | | | | | | bRop (optional) | | | | | | | | nXSrc (variable) | | | | | | | |
| ... | | | | | | | | nYSrc (variable) | | | | | | | | | | | | | | | | BackColor (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | ForeColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | BrushOrgX (optional) | | | | | | | | BrushOrgY (optional) | | | | | | | | BrushStyle (optional) | | | | | | | |
| BrushHatch (optional) | | | | | | | | BrushExtra (optional) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cacheIndex (optional) | | | | | | | | | | | | | | | |

**cacheId (2 bytes):** A 16-bit, unsigned integer. The **cacheId** field contains the encoded bitmap cache ID and Color Table Cache entry. The high byte contains the index of the color table entry to use (cached previously with a [Cache Color Table (section 2.2.2.2.1.2.4)](#Section_49fad30181714f38a68a7394baeb7b15) Secondary Drawing Order), and the low byte contains the ID of the bitmap cache in which the source bitmap is stored (cached previously with a Cache Bitmap - Revision 1 (section [2.2.2.2.1.2.2)](#Section_c9365b320be547c9af1afeba2ea1ee9f) or Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3)](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40) Secondary Drawing Order).

The color table entry MUST be in the range 0 to 5 (inclusive).

If the client advertised support for the Revision 1 bitmap caches (section [3.1.1.1.1](#Section_2BF9258842BD45278B3EB90C56E292D2)) by sending the server the Revision 1 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.1), then the cache ID MUST be in the range 0 to 2 (inclusive). However, if the client advertised support for the Revision 2 bitmap caches (section 3.1.1.1.1) by sending the server the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2), then the cache ID MUST be greater than or equal to 0 and less than the number of bitmap caches being used for the connection (specified by the **NumCellCaches** field of the Revision 2 Bitmap Cache Capability Set).

If the bitmap cache ID is set to TS\_BITMAPCACHE\_SCREEN\_ID (0xFF), the **cacheIndex** field MUST contain the index of an entry in the Offscreen Bitmap Cache that contains the source bitmap.

**nLeftRect (variable):** The left coordinate of the blit rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the blit rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nWidth (variable):** The width of the blit rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nHeight (variable):** The height of the blit rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop (1 byte):** The index of the ternary raster operation to perform (see section [2.2.2.2.1.1.1.7](#Section_a9a85075e79645ebb84af399324a1109)).

**nXSrc (variable):** The x-coordinate of the source rectangle within the source bitmap specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nYSrc (variable):** The inverted y-coordinate of the source rectangle within the source bitmap specified using a Coord Field (section 2.2.2.2.1.1.1.1). The actual value of the y-coordinate MUST be computed using the following formula:

1. ActualYSrc = (SourceBitmapHeight - nHeight) - nYSrc

**BackColor (3 bytes):** Background color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** Foreground color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BrushOrgX (1 byte):** An 8-bit, signed integer. The x-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushOrgY (1 byte):** An 8-bit, signed integer. The y-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushStyle (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushStyle** field of the [PatBlt (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8) Primary Drawing Order.

**BrushHatch (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushHatch** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**BrushExtra (7 bytes):** The contents and format of this field are the same as the **BrushExtra** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**cacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the source bitmap within the bitmap cache specified by the **cacheId** field.

If this field is set to BITMAPCACHE\_WAITING\_LIST\_INDEX (32767), the last bitmap cache entry MUST contain the bitmap data. Otherwise, this field MUST contain a value that is greater than or equal to 0 and less than the maximum number of entries allowed in the source bitmap cache. If the client advertised support for the Revision 1 bitmap caches (section 3.1.1.1.1) by sending the server the Revision 1 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.1), then the maximum number of entries allowed in each individual Revision 1 bitmap cache is specified in the Revision 1 Bitmap Cache Capability Set by the **Cache0Entries**, **Cache1Entries** and **Cache2Entries** fields. If the client advertised support for the Revision 2 bitmap caches (section 3.1.1.1.1) by sending the server the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2), then the maximum number of entries allowed in each individual Revision 2 bitmap cache is specified in the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2) by the **BitmapCache0CellInfo**, **BitmapCache1CellInfo**, **BitmapCache2CellInfo**, **BitmapCache3CellInfo** and **BitmapCache4CellInfo** fields.

If the bitmap cache ID (specified in the **cacheId** field) is set to TS\_BITMAPCACHE\_SCREEN\_ID (0xFF), this field MUST contain the index of an entry in the Offscreen Bitmap Cache that contains the source bitmap.

LineTo (LINETO\_ORDER)

The LineTo Primary Drawing Order encodes a single line drawing order that is restricted to solid color lines, one pixel wide.

1. Encoding order number: 9 (0x09)
2. Negotiation order number: 8 (0x08)
3. Number of fields: 10
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 19 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| BackMode (optional) | | | | | | | | | | | | | | | | nXStart (variable) | | | | | | | | | | | | | | | |
| nYStart (variable) | | | | | | | | | | | | | | | | nXEnd (variable) | | | | | | | | | | | | | | | |
| nYEnd (variable) | | | | | | | | | | | | | | | | BackColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | bRop2 (optional) | | | | | | | | PenStyle (optional) | | | | | | | | PenWidth (optional) | | | | | | | |
| PenColor (optional) | | | | | | | | | | | | | | | | | | | | | | | |

**BackMode (2 bytes):** An unsigned, 16-bit integer. This field contains the background mix mode and MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| TRANSPARENT  0x0001 | Background remains untouched. |
| OPAQUE  0x0002 | Background is filled with the current background color before the pen is drawn. |

**nXStart (variable):** The x-coordinate of the starting point of the line specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nYStart (variable):** The y-coordinate of the starting point of the line specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nXEnd (variable):** The x-coordinate of the ending point of the line specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nYEnd (variable):** The y-coordinate of the ending point of the line specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BackColor (3 bytes):** The background color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure. This field MUST be zeroed out.

**bRop2 (1 byte):** The binary raster operation to perform (see section [2.2.2.2.1.1.1.6](#Section_34e554e42e7c4f1fa8a6e7b319bf6fe1)).

**PenStyle (1 byte):** An 8-bit, unsigned integer. The drawing style of the pen. This field MUST be set to PS\_SOLID (0x00).

**PenWidth (1 byte):** An 8-bit, unsigned integer. The width of the pen. This field MUST be set to 0x01.

**PenColor (3 bytes):** The foreground color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

SaveBitmap (SAVEBITMAP\_ORDER)

The SaveBitmap Primary Drawing Order encodes a rectangle of the screen image for saving or restoring by the client.

1. Encoding order number: 11 (0x0B)
2. Negotiation order number: 11 (0x0B)
3. Number of fields: 6
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 13 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| SavedBitmapPosition (optional) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| nLeftRect (variable) | | | | | | | | | | | | | | | | nTopRect (variable) | | | | | | | | | | | | | | | |
| nRightRect (variable) | | | | | | | | | | | | | | | | nBottomRect (variable) | | | | | | | | | | | | | | | |
| Operation (optional) | | | | | | | |

**SavedBitmapPosition (4 bytes):** A 32-bit, unsigned integer. Encoded start position of the rectangle in the Saved Bitmap that will be read from (in the case of a bitmap restore operation) or written to (in the case of a bitmap save operation), depending on the value of the **Operation** field.

The **SavedBitmapPosition** field is constructed by using the **desktopSaveXGranularity** and **desktopSaveYGranularity** values specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3). The size of the Save Bitmap MUST be 480 x 480 pixels.

Specification [[T128]](http://go.microsoft.com/fwlink/?LinkId=90544) section 8.16.17 shows how to compute the value to insert into the **SavedBitmapPosition** field. When performing a save operation, the **SavedBitmapPosition** field contains the cumulative area of the virtual desktop rectangles already in the Save Bitmap (the area of the rectangle being written to the Save Bitmap is excluded). When performing a restore operation, the **SavedBitmapPosition** field contains the cumulative area of all the rectangles that were written to the Save Bitmap before the rectangle being restored was saved.

The functions used to compute the area by using the specified X and Y granularity are as follows.

1. AreaWidthInPixels = [(width + XGranularity - 1) / XGranularity] \* XGranularity
2. AreaHeightInPixels = [(height + YGranularity - 1) / YGranularity] \* YGranularity
3. Area = AreaWidthInPixels \* AreaHeightInPixels

To determine the X and Y position in the Save Bitmap using the YGranularity and the Save Bitmap width of 480, the following functions are used.

1. Y = [SaveBitmapPosition / (480 \* YGranularity)] \* YGranularity
2. X = [SaveBitmapPosition - (Y \* 480)] / YGranularity

An example of calculations to obtain the X and Y positions from the **SavedBitmapPosition** field are defined visually in section [4.5](#Section_5337b5e437084570a6821791ffc90c88).

**nLeftRect (variable):** The left coordinate of the virtual desktop rectangle to save specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**nTopRect (variable):** The top coordinate of the virtual desktop rectangle to save specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nRightRect (variable):** The right inclusive coordinate of the virtual desktop rectangle to save specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**nBottomRect (variable):** The bottom inclusive coordinate of the virtual desktop rectangle to save specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**Operation (1 byte):** An 8-bit, unsigned integer. The operation to perform that MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SV\_SAVEBITS  0x00 | Save bitmap operation. |
| SV\_RESTOREBITS  0x01 | Restore bitmap operation. |

GlyphIndex (GLYPHINDEX\_ORDER)

The GlyphIndex Primary Drawing Order encodes a set of glyph indices at a specified position.

1. Encoding order number: 27 (0x1B)
2. Negotiation order number: 27 (0x1B)
3. Number of fields: 22
4. Number of field encoding bytes: 3
5. Maximum encoded field length: 297 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId (optional) | | | | | | | | flAccel (optional) | | | | | | | | ulCharInc (optional) | | | | | | | | fOpRedundant (optional) | | | | | | | |
| BackColor (optional) | | | | | | | | | | | | | | | | | | | | | | | | ForeColor (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | BkLeft (optional) | | | | | | | | | | | | | | | |
| BkTop (optional) | | | | | | | | | | | | | | | | BkRight (optional) | | | | | | | | | | | | | | | |
| BkBottom (optional) | | | | | | | | | | | | | | | | OpLeft (optional) | | | | | | | | | | | | | | | |
| OpTop (optional) | | | | | | | | | | | | | | | | OpRight (optional) | | | | | | | | | | | | | | | |
| OpBottom (optional) | | | | | | | | | | | | | | | | BrushOrgX (optional) | | | | | | | | BrushOrgY (optional) | | | | | | | |
| BrushStyle (optional) | | | | | | | | BrushHatch (optional) | | | | | | | | BrushExtra (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | X (optional) | | | | | | | | | | | | | | | | Y (optional) | | | | | | | |
| ... | | | | | | | | VariableBytes (variable) | | | | | | | | | | | | | | | |

**cacheId (1 byte):** An 8-bit, unsigned integer. The ID of the glyph cache in which the glyph data MUST be stored. This value MUST be in the range 0 to 9 (inclusive).

**flAccel (1 byte):** An 8-bit, unsigned integer. Accelerator flags. For glyph related terminology, see [YUAN] figures 14-17 and 15-1. For information about string widths and heights, see [[MSDN-SWH]](http://go.microsoft.com/fwlink/?LinkId=196875). For information about character widths, see [[MSDN-CW]](http://go.microsoft.com/fwlink/?LinkId=196874). This field MUST contain a combination of the following flags.

| Value | Meaning |
| --- | --- |
| SO\_FLAG\_DEFAULT\_PLACEMENT  0x01 | This flag MUST be set. |
| SO\_HORIZONTAL  0x02 | Text is horizontal, left-to-right or right-to-left, depending on SO\_REVERSED. |
| SO\_VERTICAL  0x04 | Text is vertical, top-to-bottom or bottom-to-top, depending on SO\_REVERSED. |
| SO\_REVERSED  0x08 | Set if horizontal text is right-to-left or vertical text is bottom-to-top. |
| SO\_ZERO\_BEARINGS  0x10 | For a given glyph in the font, the A-width (left-side bearing) and C-width (right-side bearing) associated with the glyph have a value of zero. |
| SO\_CHAR\_INC\_EQUAL\_BM\_BASE  0x20 | For a given glyph in the font, the B-width associated with the glyph equals the advance width of the glyph. |
| SO\_MAXEXT\_EQUAL\_BM\_SIDE  0x40 | The height of the bitmap associated with a given glyph in the font is always equal to the sum of the ascent and descent. This implies that the tops and bottoms of all glyph bitmaps lie on the same line in the direction of writing. |

**ulCharInc (1 byte):** An 8-bit, unsigned integer. Specifies whether or not the font is a fixed-pitch (monospace) font. If so, this member is equal to the advance width of the glyphs in pixels (see [YUAN] figures 14-17); if not, this field is set to 0x00. The minimum value for this field is 0x00 (inclusive), and the maximum value is 0xFF (inclusive).

**fOpRedundant (1 byte):** An 8-bit, unsigned integer. A Boolean value indicating whether or not the opaque rectangle is redundant. Redundant, in this context, means that the text background is transparent.

| Value | Meaning |
| --- | --- |
| FALSE  0x00 | Rectangle is not redundant. |
| TRUE  0x01 | Rectangle is redundant. |

**BackColor (3 bytes):** The text color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** Color of the opaque rectangle described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BkLeft (2 bytes):** A 16-bit, signed integer. The left coordinate of the text background rectangle.

**BkTop (2 bytes):** A 16-bit, signed integer. The top coordinate of the text background rectangle.

**BkRight (2 bytes):** A 16-bit, signed integer. The right coordinate of the text background rectangle.

**BkBottom (2 bytes):** A 16-bit, signed integer. The bottom coordinate of the text background rectangle.

**OpLeft (2 bytes):** A 16-bit, signed integer. The left coordinate of the opaque rectangle. This field MUST be set to 0 if the fOpRedundant flag is set.

**OpTop (2 bytes):** A 16-bit, signed integer. The top coordinate of the opaque rectangle. This field MUST be set to 0 if the fOpRedundant flag is set.

**OpRight (2 bytes):** A 16-bit, signed integer. The right coordinate of the opaque rectangle. This field MUST be set to 0 if the fOpRedundant flag is set.

**OpBottom (2 bytes):** A 16-bit, signed integer. The bottom coordinate of the opaque rectangle. This field MUST be set to 0 if the fOpRedundant flag is set.

**BrushOrgX (1 byte):** An 8-bit, signed integer. The x-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushOrgY (1 byte):** An 8-bit, signed integer. The y-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushStyle (1 byte):** An 8-bit, unsigned integer. This field MUST be set to BS\_SOLID (0x00), as the GlyphIndex Primary Drawing Order MUST only use solid color brushes to render the opaque rectangle.

**BrushHatch (1 byte):** An 8-bit, unsigned integer. This field MUST be set to 0x00, as the GlyphIndex Primary Drawing Order MUST only use solid color brushes to render the opaque rectangle.

**BrushExtra (7 bytes):** This field is not used, as the GlyphIndex Primary Drawing Order MUST only use solid color brushes to render the opaque rectangle.

**X (2 bytes):** A 16-bit, signed integer. The x-coordinate of the point where the origin of the starting glyph MUST be positioned.

**Y (2 bytes):** A 16-bit, signed integer. The y-coordinate of the point where the origin of the starting glyph MUST be positioned.

**VariableBytes (variable):** A [One-Byte Header Variable Field (section 2.2.2.2.1.1.1.2)](#Section_008cb2338ae94e05ad0383f284f20a61) structure. This field MUST contain glyph fragments (which are composed of a series of one or more glyph cache indices) and instructions to use entries previously stored in the glyph fragment cache. Multiple glyph fragments can be contained in this field. The first byte of each fragment is either a USE (0xFE) operation byte or a glyph index (0x00 to 0x0FD) byte:

* A value of 0xFE (USE) indicates that a previously stored fragment MUST be displayed. The byte following the USE byte is the index in the fragment cache where the fragment is located. This fragment MUST be read and displayed. If the **ulCharInc** field is set to 0 and the **flAccel** field does not contain the SO\_CHAR\_INC\_EQUAL\_BM\_BASE (0x20) flag, then the index byte MUST be followed by a delta byte that indicates the distance between two consecutive fragments; this distance is measured in pixels from the beginning of the first fragment to the beginning of the next. If the distance is greater than 127 (0x7F), then the value 0x80 MUST be used, and the following two bytes will be set to contain the actual distance formatted as an unsigned integer in little-endian order.
* If not preceded by 0xFE, a value of 0x00 to 0xFD identifies a glyph stored at the given index in the glyph cache. Multiple glyphs can be sent at one time. If the **ulCharInc** field is set to 0 and the **flAccel** field does not contain the SO\_CHAR\_INC\_EQUAL\_BM\_BASE (0x20) flag, then the index byte MUST be followed by a delta byte that indicates the distance between two consecutive glyphs; this distance is measured in pixels from the beginning of the first glyph to the beginning of the next. If the distance is greater than 127 (0x7F), then the value 0x80 MUST be used, and the following two bytes will be set to contain the actual distance formatted as an unsigned integer in little-endian order.

If a series of glyph indices ends with an ADD (0xFF) operation byte, the preceding glyph information MUST be collected, displayed, and then stored in the fragment cache. The byte following the ADD byte is the index of the cache where the fragment MUST be stored. A final byte that indicates the size of the fragment follows the index byte. (The ADD byte, index byte, and size byte MUST NOT be counted when calculating the value of the size byte.)

All glyph cache indices MUST be greater than or equal to 0, and less than the maximum number of entries allowed in the glyph cache with the ID specified by the **cacheId** field. The maximum number of entries allowed in each of the ten glyph caches is specified in the **GlyphCache** field of the Glyph Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

All fragment cache indices MUST be in the range 0 to 255 (inclusive).

FastIndex (FASTINDEX\_ORDER)

The FastIndex Primary Drawing Order encodes a set of glyph indices at a specified position. This order is an improved version of the [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) Primary Drawing Order. The regular GlyphIndex order contains five brush fields that the FastIndex order does not: **BrushOrgX**, **BrushOrgY**, **BrushStyle**, **BrushHatch**, and **BrushExtra**. These extra fields MUST all be implicitly assumed to exist and contain default values of zero (implying that a solid color brush will be used).

1. Encoding order number: 19 (0x13)
2. Negotiation order number: 19 (0x13)
3. Number of fields: 15
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 285 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId (optional) | | | | | | | | fDrawing (optional) | | | | | | | | | | | | | | | | BackColor (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | ForeColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | BkLeft (variable) | | | | | | | | | | | | | | | | BkTop (variable) | | | | | | | |
| ... | | | | | | | | BkRight (variable) | | | | | | | | | | | | | | | | BkBottom (variable) | | | | | | | |
| ... | | | | | | | | OpLeft (variable) | | | | | | | | | | | | | | | | OpTop (variable) | | | | | | | |
| ... | | | | | | | | OpRight (variable) | | | | | | | | | | | | | | | | OpBottom (variable) | | | | | | | |
| ... | | | | | | | | X (optional) | | | | | | | | | | | | | | | | Y (optional) | | | | | | | |
| ... | | | | | | | | VariableBytes (variable) | | | | | | | | | | | | | | | |

**cacheId (1 byte):** An 8-bit, unsigned integer. The ID of the glyph cache in which the glyph data MUST be stored. This value MUST be in the range 0 to 9 (inclusive).

**fDrawing (2 bytes):** A 16-bit, unsigned integer. Combined **flAccel** and **ulCharInc** fields from the GlyphIndex (section 2.2.2.2.1.1.2.13) Primary Drawing Order. The high-order byte contains the **flAccel** field, and the low-order byte contains the **ulCharInc** field.

**BackColor (3 bytes):** The text color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** Opaque rectangle color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BkLeft (variable):** Left coordinate of the text background rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**BkTop (variable):** Top coordinate of the text background rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BkRight (variable):** Right coordinate of the text background rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BkBottom (variable):** Bottom coordinate of the text background rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**OpLeft (variable):** Left coordinate of the opaque rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to 0 if it is the same as **BkLeft**.

**OpTop (variable):** The top coordinate of the opaque rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

This field MUST contain opaque rectangle encoding flags if the **OpBottom** field is set to be the value of -32768. In this case, **OpTop** holds the encoding flags in the low 4 bits.

| Value | Meaning |
| --- | --- |
| OPRECT\_BOTTOM\_ABSENT  0x01 | Bottom coordinate of the opaque rectangle is the same as **BkBottom**. |
| OPRECT\_RIGHT\_ABSENT  0x02 | Right coordinate of the opaque rectangle is the same as **BkRight**. |
| OPRECT\_TOP\_ABSENT  0x04 | Top coordinate of the opaque rectangle is the same as **BkTop**. |
| OPRECT\_LEFT\_ABSENT  0x08 | Left coordinate of the opaque rectangle is the same as **BkLeft**. |

The only valid combinations of the encoding flags that are currently supported are as follows:

* 0x0F: The left, top, right, and bottom coordinates of the opaque rectangle all match the background text rectangle and are absent.
* 0x0D: The left, top, and bottom coordinates of the opaque rectangle all match the background text rectangle and are absent. The actual value of the right coordinate is present.

**OpRight (variable):** The right coordinate of the opaque rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to 0 if it is the same as **BkRight**.

**OpBottom (variable):** The bottom coordinate of the opaque rectangle specified using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to -32768 if **OpTop** contains opaque rectangle encoding flags (see the **OpTop** field).

**X (2 bytes):** A 16-bit, signed integer. The x-coordinate of the point where the origin of the starting glyph MUST be positioned specified using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to -32768 if it is the same as **BkLeft**.

**Y (2 bytes):** A 16-bit, signed integer. The y-coordinate of the point where the origin of the starting glyph MUST be positioned specified using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to -32768 if it is the same as **BkTop**.

**VariableBytes (variable):** A [One-Byte Header Variable Field (section 2.2.2.2.1.1.1.2)](#Section_008cb2338ae94e05ad0383f284f20a61) structure. The contents and format of this field are the same as the **VariableBytes** field of the GlyphIndex (section 2.2.2.2.1.1.2.13) Primary Drawing Order.

FastGlyph (FASTGLYPH\_ORDER)

The FastGlyph Primary Drawing Order encodes a single glyph at a specified position. This primary drawing order is a fast way of outputting a single glyph and bypasses having to send a Cache Glyph Secondary Drawing Order (see sections [2.2.2.2.1.2.5](#Section_6dc06088b1124f4fb0bcb97bba621a0c) and [2.2.2.2.1.2.6](#Section_ffc80435136e442e908540ca1d27a413)) followed by a [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) or [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7) Primary Drawing Order.

1. Encoding order number: 24 (0x18)
2. Negotiation order number: 24 (0x18)
3. Number of fields: 15
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 285 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId (optional) | | | | | | | | fDrawing (optional) | | | | | | | | | | | | | | | | BackColor (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | ForeColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | BkLeft (variable) | | | | | | | | | | | | | | | | BkTop (variable) | | | | | | | |
| ... | | | | | | | | BkRight (variable) | | | | | | | | | | | | | | | | BkBottom (variable) | | | | | | | |
| ... | | | | | | | | OpLeft (variable) | | | | | | | | | | | | | | | | OpTop (variable) | | | | | | | |
| ... | | | | | | | | OpRight (variable) | | | | | | | | | | | | | | | | OpBottom (variable) | | | | | | | |
| ... | | | | | | | | X (variable) | | | | | | | | | | | | | | | | Y (variable) | | | | | | | |
| ... | | | | | | | | VariableBytes (variable) | | | | | | | | | | | | | | | |

**cacheId (1 byte):** An 8-bit, unsigned integer. The ID of the glyph cache in which the glyph data MUST be stored. This value MUST be in the range 0 to 9 (inclusive).

**fDrawing (2 bytes):** A 16-bit, unsigned integer. Combined **flAccel** and **ulCharInc** fields from the GlyphIndex (section 2.2.2.2.1.1.2.13) Primary Drawing Order. The high-order byte contains the **flAccel** field, and the low-order byte contains the **ulCharInc** field.

**BackColor (3 bytes):** Text color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** The opaque rectangle color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BkLeft (variable):** The left coordinate of the text background rectangle specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**BkTop (variable):** The top coordinate of the text background rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BkRight (variable):** The right coordinate of the text background rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BkBottom (variable):** The bottom coordinate of the text background rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**OpLeft (variable):** The left coordinate of the opaque rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to 0 if it is the same as **BkLeft**.

**OpTop (variable):** The top coordinate of the opaque rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST contain opaque rectangle encoding flags if the **OpBottom** field is set to -32768. For details on the opaque rectangle encoding flags, see the **OpTop** field of the FastIndex (section 2.2.2.2.1.1.2.14) Primary Drawing Order.

**OpRight (variable):** The right coordinate of the opaque rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to 0 if it is the same as **BkRight**.

**OpBottom (variable):** The bottom coordinate of the opaque rectangle specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to -32768 if **OpTop** contains opaque rectangle encoding flags (see the **OpTop** field).

**X (variable):** A 16-bit, signed integer. The x-coordinate of the point where the origin of the starting glyph MUST be positioned, specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to -32768 if it is the same as **BkLeft**.

**Y (variable):** A 16-bit, signed integer. The y-coordinate of the point where the origin of the starting glyph MUST be positioned, specified by using a Coord Field (section 2.2.2.2.1.1.1.1). This field MUST be set to -32768 if it is the same as **BkTop**.

**VariableBytes (variable):** A [One-Byte Header Variable Field (section 2.2.2.2.1.1.1.2)](#Section_008cb2338ae94e05ad0383f284f20a61) structure. If the size of this **VariableBytes** field is 1 byte, it contains a 1-byte glyph cache index from which the glyph data MUST be retrieved. However, if the size is larger than 1 byte, this field contains a Cache Glyph Data - Revision 2 (section [2.2.2.2.1.2.6.1)](#Section_7facfe557db14a47a38ebc223cce5dc4) structure. The glyph data MUST be stored in the glyph cache specified by the **cacheId** field in the entry indicated by the **cacheIndex** field of the glyph data. The Cache Glyph Data - Revision 2 structure MUST be followed by 2 bytes of padding that MAY[<2>](#Appendix_A_2) contain a little-endian encoded [**Unicode character**](#gt_fd33af2e-e1ce-4f8e-a706-f9fb8123f9b0) representing the glyph (this Unicode character MUST NOT be null-terminated).

All glyph cache indices MUST be greater than or equal to 0, and less than the maximum number of entries allowed in the glyph cache with the ID specified by the **cacheId** field. The maximum number of entries allowed in each of the ten glyph caches is specified in the **GlyphCache** field of the Glyph Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

PolygonSC (POLYGON\_SC\_ORDER)

The PolygonSC Primary Drawing Order encodes a solid-color polygon consisting of two or more vertices connected by straight lines.

1. Encoding order number: 20 (0x14)
2. Negotiation order number: 20 (0x14)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 249 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| xStart (variable) | | | | | | | | | | | | | | | | yStart (variable) | | | | | | | | | | | | | | | |
| bRop2 (optional) | | | | | | | | FillMode (optional) | | | | | | | | BrushColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | NumDeltaEntries (optional) | | | | | | | | CodedDeltaList (variable) | | | | | | | | | | | | | | | |

**xStart (variable):** The x-coordinate of the starting point of the polygon path specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**yStart (variable):** The y-coordinate of the starting point of the polygon path specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop2 (1 byte):** The binary raster operation to perform (see section [2.2.2.2.1.1.1.6](#Section_34e554e42e7c4f1fa8a6e7b319bf6fe1)).

**FillMode (1 byte):** The polygon filling algorithm described using a [Fill Mode (section 2.2.2.2.1.1.1.9)](#Section_06dc707471cd41cf9028964877875dfc) structure.

**BrushColor (3 bytes):** Foreground color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**NumDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of points along the polygon path described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [One-Byte Header Variable Field (section 2.2.2.2.1.1.1.2)](#Section_008cb2338ae94e05ad0383f284f20a61) structure that encapsulates a [Delta-Encoded Points (section 2.2.2.2.1.1.1.4)](#Section_6c7b2a52103c4a7da2a9997416d4a475) structure that contains the points along the polygon path. The number of points described by the Delta-Encoded Points structure is specified by the **NumDeltaEntries** field.

PolygonCB (POLYGON\_CB\_ORDER)

The PolygonCB Primary Drawing Order encodes a color brush polygon consisting of two or more vertices connected by straight lines.

1. Encoding order number: 21 (0x15)
2. Negotiation order number: 21 (0x15)
3. Number of fields: 13
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 263 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| xStart (variable) | | | | | | | | | | | | | | | | yStart (variable) | | | | | | | | | | | | | | | |
| bRop2 (optional) | | | | | | | | FillMode (optional) | | | | | | | | BackColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | ForeColor (optional) | | | | | | | | | | | | | | | | | | | | | | | |
| BrushOrgX (optional) | | | | | | | | BrushOrgY (optional) | | | | | | | | BrushStyle (optional) | | | | | | | | BrushHatch (optional) | | | | | | | |
| BrushExtra (optional) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | NumDeltaEntries (optional) | | | | | | | |
| CodedDeltaList (variable) | | | | | | | | | | | | | | | |

**xStart (variable):** The x-coordinate of the starting point of the polygon path specified using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**yStart (variable):** The y-coordinate of the starting point of the polygon path specified using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop2 (1 byte):** The binary raster operation to perform (see section [2.2.2.2.1.1.1.6](#Section_34e554e42e7c4f1fa8a6e7b319bf6fe1)). The ROP2 field for the PolygonCB Primary Drawing Order has two bitfields within it. The low 5 bits (mask 0x1F) identify the real ROP2 operation. The high bit (mask 0x80) indicates whether the background drawing mode MUST be set to TRANSPARENT or OPAQUE (see section [2.2.2.2.1.1.2.11](#Section_27276fcf32be49a99a1797ec6167ac7e)). The background drawing mode is only significant if the **BrushStyle** field indicates that the brush is a BS\_HATCHED (0x02) or BS\_PATTERN (0x03) brush.

**FillMode (1 byte):** The polygon filling algorithm described by using a [Fill Mode (section 2.2.2.2.1.1.1.9)](#Section_06dc707471cd41cf9028964877875dfc) structure.

**BackColor (3 bytes):** The background color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** The foreground color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BrushOrgX (1 byte):** An 8-bit, signed integer. The x-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushOrgY (1 byte):** An 8-bit, signed integer. The y-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushStyle (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushStyle** field of the [PatBlt (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8) Primary Drawing Order.

**BrushHatch (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushHatch** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**BrushExtra (7 bytes):** A byte array of length 7. The contents and format of this field are the same as the **BrushExtra** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**NumDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of points along the polygon path described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [One-Byte Header Variable Field (section 2.2.2.2.1.1.1.2)](#Section_008cb2338ae94e05ad0383f284f20a61) structure that encapsulates a [Delta-Encoded Points (section 2.2.2.2.1.1.1.4)](#Section_6c7b2a52103c4a7da2a9997416d4a475) structure that contains the points along the polygon path. The number of points described by the Delta-Encoded Points structure is specified by the **NumDeltaEntries** field.

Polyline (POLYLINE\_ORDER)

The Polyline Primary Drawing Order encodes a solid color polyline consisting of two or more vertices connected by straight lines.

1. Encoding order number: 22 (0x16)
2. Negotiation order number: 22 (0x16)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 148 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| xStart (variable) | | | | | | | | | | | | | | | | yStart (variable) | | | | | | | | | | | | | | | |
| bRop2 (optional) | | | | | | | | BrushCacheEntry (optional) | | | | | | | | | | | | | | | | PenColor (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | NumDeltaEntries (optional) | | | | | | | | CodedDeltaList (variable) | | | | | | | |
| ... | | | | | | | |

**xStart (variable):** The x-coordinate of the starting point of the polygon path specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**yStart (variable):** The y-coordinate of the starting point of the polygon path specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop2 (1 byte):** The binary raster operation to perform (see section [2.2.2.2.1.1.1.6](#Section_34e554e42e7c4f1fa8a6e7b319bf6fe1)).

**BrushCacheEntry (2 bytes):** A 16-bit unsigned integer. The brush cache entry. This field is unused (as only solid color polylines are drawn) and SHOULD[<3>](#Appendix_A_3) be set to 0x0000.

**PenColor (3 bytes):** The foreground color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**NumDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of points along the polyline path described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [One-Byte Header Variable Field (section 2.2.2.2.1.1.1.2)](#Section_008cb2338ae94e05ad0383f284f20a61) structure that encapsulates a [Delta-Encoded Points (section 2.2.2.2.1.1.1.4)](#Section_6c7b2a52103c4a7da2a9997416d4a475) structure that contains the points along the polyline path. The number of points described by the Delta-Encoded Points structure is specified by the **NumDeltaEntries** field.

EllipseSC (ELLIPSE\_SC\_ORDER)

The EllipseSC Primary Drawing Order encodes a single, solid-color ellipse.

1. Encoding order number: 25 (0x19)
2. Negotiation order number: 25 (0x19)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 13 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| LeftRect (variable) | | | | | | | | | | | | | | | | TopRect (variable) | | | | | | | | | | | | | | | |
| RightRect (variable) | | | | | | | | | | | | | | | | BottomRect (variable) | | | | | | | | | | | | | | | |
| bRop2 (optional) | | | | | | | | FillMode (optional) | | | | | | | | Color (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**LeftRect (variable):** The left coordinate of the inclusive rectangle for the ellipse specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**TopRect (variable):** The top coordinate of the inclusive rectangle for the ellipse specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**RightRect (variable):** The right coordinate of the inclusive rectangle for the ellipse specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BottomRect (variable):** The bottom coordinate of the inclusive rectangle for the ellipse specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop2 (1 byte):** The binary raster operation to perform (see section [2.2.2.2.1.1.1.6](#Section_34e554e42e7c4f1fa8a6e7b319bf6fe1)).

**FillMode (1 byte):** An 8-bit, unsigned integer that specifies the fill mode. **FillMode** MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| NOFILLL  0x00 | A polyline ellipse (that is, a non-filled ellipse) MUST be drawn. |
| ALTERNATE  0x01 | See section [2.2.2.2.1.1.1.9](#Section_06dc707471cd41cf9028964877875dfc) for an explanation of this value. |
| WINDING  0x02 | See section 2.2.2.2.1.1.1.9 for an explanation of this value. |

**Color (3 bytes):** The foreground color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

EllipseCB (ELLIPSE\_CB\_ORDER)

The EllipseCB Primary Drawing Order encodes a color brush ellipse.

1. Encoding order number: 26 (0x1A)
2. Negotiation order number: 26 (0x1A)
3. Number of fields: 13
4. Number of field encoding bytes: 2
5. Maximum encoded field length: 27 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| LeftRect (variable) | | | | | | | | | | | | | | | | TopRect (variable) | | | | | | | | | | | | | | | |
| RightRect (variable) | | | | | | | | | | | | | | | | BottomRect (variable) | | | | | | | | | | | | | | | |
| bRop2 (optional) | | | | | | | | FillMode (optional) | | | | | | | | BackColor (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | ForeColor (optional) | | | | | | | | | | | | | | | | | | | | | | | |
| BrushOrgX (optional) | | | | | | | | BrushOrgY (optional) | | | | | | | | BrushStyle (optional) | | | | | | | | BrushHatch (optional) | | | | | | | |
| BrushExtra (optional) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | |

**LeftRect (variable):** The left coordinate of the inclusive rectangle for the ellipse specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**TopRect (variable):** The top coordinate of the inclusive rectangle for the ellipse specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**RightRect (variable):** The right coordinate of the inclusive rectangle for the ellipse specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**BottomRect (variable):** The bottom coordinate of the inclusive rectangle for the ellipse specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bRop2 (1 byte):** The binary raster operation to perform (see section [2.2.2.2.1.1.1.6](#Section_34e554e42e7c4f1fa8a6e7b319bf6fe1)). The ROP2 field for the EllipseCB Primary Drawing Order has two bitfields within it. The low 5 bits (mask 0x1F) identify the real ROP2 operation. The high bit (mask 0x80) indicates whether the background drawing mode MUST be set to TRANSPARENT or OPAQUE (see section [2.2.2.2.1.1.2.11](#Section_27276fcf32be49a99a1797ec6167ac7e)). The background drawing mode is only significant if the **BrushStyle** field indicates that the brush is a BS\_HATCHED (0x02) or BS\_PATTERN (0x03) brush.

**FillMode (1 byte):** The ellipse filling algorithm described by using a [Fill Mode (section 2.2.2.2.1.1.1.9)](#Section_06dc707471cd41cf9028964877875dfc) structure.

**BackColor (3 bytes):** The background color described by using a [Generic Color (section 2.2.2.2.1.1.1.8)](#Section_f99c616b876d437199fecc656d0a610b) structure.

**ForeColor (3 bytes):** The foreground color described by using a Generic Color (section 2.2.2.2.1.1.1.8) structure.

**BrushOrgX (1 byte):** An 8-bit, signed integer. The x-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushOrgY (1 byte):** An 8-bit, signed integer. The y-coordinate of the point where the top leftmost pixel of a brush pattern MUST be anchored.

**BrushStyle (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushStyle** field of the [PatBlt (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8) Primary Drawing Order.

**BrushHatch (1 byte):** An 8-bit, unsigned integer. The contents and format of this field are the same as the **BrushHatch** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

**BrushExtra (7 bytes):** A byte array of length 7. The contents and format of this field are the same as the **BrushExtra** field of the PatBlt (section 2.2.2.2.1.1.2.3) Primary Drawing Order.

DrawNineGrid (DRAWNINEGRID\_ORDER)

The DrawNineGrid Primary Drawing Order encodes a single NineGrid drawing command with a single bounding rectangle.

1. Encoding order number: 7 (0x07)
2. Negotiation order number: 7 (0x07)
3. Number of fields: 5
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 10 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| srcLeft (variable) | | | | | | | | | | | | | | | | srcTop (variable) | | | | | | | | | | | | | | | |
| srcRight (variable) | | | | | | | | | | | | | | | | srcBottom (variable) | | | | | | | | | | | | | | | |
| bitmapId | | | | | | | | | | | | | | | |

**srcLeft (variable):** The left coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**srcTop (variable):** The top coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**srcRight (variable):** The right coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**srcBottom (variable):** The bottom coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bitmapId (2 bytes):** A 16-bit, unsigned integer. The index of the NineGrid Bitmap Cache entry wherein the bitmap and NineGrid transformation information are stored. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the NineGrid Bitmap Cache as specified by the **drawNineGridCacheEntries** field of the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780). The bitmap and transformation information stored in the cache MUST have already been cached in response to a [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38) Alternate Secondary Drawing Order.

MultiDrawNineGrid (MULTI\_DRAWNINEGRID\_ORDER)

The MultiDrawNineGrid Primary Drawing Order encodes a single NineGrid drawing command with multiple clipping rectangles.

1. Encoding order number: 8 (0x08)
2. Negotiation order number: 9 (0x09)
3. Number of fields: 7
4. Number of field encoding bytes: 1
5. Maximum encoded field length: 396 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| srcLeft (variable) | | | | | | | | | | | | | | | | srcTop (variable) | | | | | | | | | | | | | | | |
| srcRight (variable) | | | | | | | | | | | | | | | | srcBottom (variable) | | | | | | | | | | | | | | | |
| bitmapId (optional) | | | | | | | | | | | | | | | | nDeltaEntries (optional) | | | | | | | | CodedDeltaList (variable) | | | | | | | |
| ... | | | | | | | |

**srcLeft (variable):** The left coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a [Coord Field (section 2.2.2.2.1.1.1.1)](#Section_870b982c9abe40cb99d02c3f0cc5fb74).

**srcTop (variable):** The top coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**srcRight (variable):** The right coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**srcBottom (variable):** The bottom coordinate of the clipping rectangle to be applied to the bitmap stored at the entry given by the **bitmapId** field. The coordinate is specified by using a Coord Field (section 2.2.2.2.1.1.1.1).

**bitmapId (2 bytes):** A 16-bit, unsigned integer. The index of the NineGrid Bitmap Cache entry wherein the bitmap and NineGrid transformation information are stored. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the NineGrid Bitmap Cache as specified by the **drawNineGridCacheEntries** field of the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780). The bitmap and transformation information stored in the cache MUST have already been cached in response to a [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38) Alternate Secondary Drawing Order.

**nDeltaEntries (1 byte):** An 8-bit, unsigned integer. The number of bounding rectangles described by the **CodedDeltaList** field.

**CodedDeltaList (variable):** A [Two-Byte Header Variable Field (section 2.2.2.2.1.1.1.3)](#Section_aead742ea85847f1a7790311660f84af) structure that encapsulates a [Delta-Encoded Rectangles (section 2.2.2.2.1.1.1.5)](#Section_b89f2058b1804da09bd1aa694c87768c) structure that contains bounding rectangles to use when rendering the order. The number of rectangles described by the Delta-Encoded Rectangles structure is specified by the **nDeltaEntries** field.

###### Secondary Drawing Orders

Common Data Types

Secondary Drawing Order Header (SECONDARY\_DRAWING\_ORDER\_HEADER)

The SECONDARY\_DRAWING\_ORDER\_HEADER structure is included in all secondary drawing orders.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| controlFlags | | | | | | | | orderLength | | | | | | | | | | | | | | | | extraFlags | | | | | | | |
| ... | | | | | | | | orderType | | | | | | | |

**controlFlags (1 byte):** An 8-bit, unsigned integer. The control byte that identifies the class of the drawing order. This field MUST contain the TS\_STANDARD (0x01) and TS\_SECONDARY (0x02) flags to indicate that the order is a secondary drawing order (see section [2.2.2.1.1](#Section_762f3dd36a76485786a4a3ca951ca3a6)).

**orderLength (2 bytes):** A 16-bit, signed integer. The encoded length in bytes of the secondary drawing order, including the size of the header. When constructing the order, the value in the **orderLength** field MUST be 13 bytes less than the actual order length. Hence, when decoding the order, the **orderLength** field MUST be adjusted by adding 13 bytes. These adjustments are for historical reasons.

**extraFlags (2 bytes):** A 16-bit, unsigned integer. Flags specific to each secondary drawing order.

**orderType (1 byte):** An 8-bit, unsigned integer. Identifies the type of secondary drawing order. MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| TS\_CACHE\_BITMAP\_UNCOMPRESSED  0x00 | Cache Bitmap - Revision 1 (section [2.2.2.2.1.2.2)](#Section_c9365b320be547c9af1afeba2ea1ee9f) Secondary Drawing Order with an uncompressed bitmap. |
| TS\_CACHE\_COLOR\_TABLE  0x01 | [Cache Color Table (section 2.2.2.2.1.2.4)](#Section_49fad30181714f38a68a7394baeb7b15) Secondary Drawing Order. |
| TS\_CACHE\_BITMAP\_COMPRESSED  0x02 | Cache Bitmap - Revision 1 (section 2.2.2.2.1.2.2) Secondary Drawing Order with a compressed bitmap. |
| TS\_CACHE\_GLYPH  0x03 | Cache Glyph - Revision 1 (section [2.2.2.2.1.2.5)](#Section_6dc06088b1124f4fb0bcb97bba621a0c) or Cache Glyph - Revision 2 (section [2.2.2.2.1.2.6)](#Section_ffc80435136e442e908540ca1d27a413) Secondary Drawing Order. The version is indicated by the extraFlags field. |
| TS\_CACHE\_BITMAP\_UNCOMPRESSED\_REV2  0x04 | Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3)](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40) Secondary Drawing Order with an uncompressed bitmap. |
| TS\_CACHE\_BITMAP\_COMPRESSED\_REV2  0x05 | Cache Bitmap - Revision 2 (section 2.2.2.2.1.2.3) Secondary Drawing Order with a compressed bitmap. |
| TS\_CACHE\_BRUSH  0x07 | [Cache Brush (section 2.2.2.2.1.2.7)](#Section_9e7027c4d9894cb09674e2e36c95ebdd) Secondary Drawing Order. |
| TS\_CACHE\_BITMAP\_COMPRESSED\_REV3  0x08 | Cache Bitmap - Revision 3 (section [2.2.2.2.1.2.8)](#Section_710cf5c26a0c4cea8d62fa4f639d7c5a) Secondary Drawing Order with a compressed bitmap. |

Two-Byte Unsigned Encoding (TWO\_BYTE\_UNSIGNED\_ENCODING)

The TWO\_BYTE\_UNSIGNED\_ENCODING structure is used to encode a value in the range 0x0000 to 0x7FFF by using a variable number of bytes. For example, 0x1A1B is encoded as { 0x9A, 0x1B }. The most significant bit of the first byte encodes the number of bytes in the structure.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| c | val1 | | | | | | | val2 (optional) | | | | | | | |

**c (1 bit):** A 1-bit, unsigned integer field that contains an encoded representation of the number of bytes in this structure.

| Value | Meaning |
| --- | --- |
| ONE\_BYTE\_VAL  0 | Implies that the optional **val2** field is not present. Hence, the structure is 1 byte in size. |
| TWO\_BYTE\_VAL  1 | Implies that the optional **val2** field is present. Hence, the structure is 2 bytes in size. |

**val1 (7 bits):** A 7-bit, unsigned integer field containing the most significant 7 bits of the value represented by this structure.

**val2 (1 byte):** An 8-bit, unsigned integer containing the least significant bits of the value represented by this structure.

Two-Byte Signed Encoding (TWO\_BYTE\_SIGNED\_ENCODING)

The TWO\_BYTE\_SIGNED\_ENCODING structure is used to encode a value in the range -0x3FFF to 0x3FFF by using a variable number of bytes. For example, -0x1A1B is encoded as { 0xDA, 0x1B }, and -0x0002 is encoded as { 0x42 }. The most significant bits of the first byte encode the number of bytes in the structure and the sign.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| c | s | val1 | | | | | | val2 (optional) | | | | | | | |

**c (1 bit):** A 1-bit, unsigned integer field containing an encoded representation of the number of bytes in this structure.

| Value | Meaning |
| --- | --- |
| ONE\_BYTE\_VAL  0 | Implies that the optional **val2** field is not present. Hence, the structure is 1 byte in size. |
| TWO\_BYTE\_VAL  1 | Implies that the optional **val2** field is present. Hence, the structure is 2 bytes in size. |

**s (1 bit):** A 1-bit, unsigned integer field containing an encoded representation of whether the value is positive or negative.

| Value | Meaning |
| --- | --- |
| POSITIVE\_VAL  0 | Implies that the value represented by this structure is positive. |
| NEGATIVE\_VAL  1 | Implies that the value represented by this structure is negative. |

**val1 (6 bits):** A 6-bit, unsigned integer field containing the most significant 6 bits of the value represented by this structure.

**val2 (1 byte):** An 8-bit, unsigned integer containing the least significant bits of the value represented by this structure.

Four-Byte Unsigned Encoding (FOUR\_BYTE\_UNSIGNED\_ENCODING)

The FOUR\_BYTE\_UNSIGNED\_ENCODING structure is used to encode a value in the range 0x00000000 to 0x3FFFFFFF by using a variable number of bytes. For example, 0x001A1B1C is encoded as { 0x9A, 0x1B, 0x1C }. The two most significant bits of the first byte encode the number of bytes in the structure.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| c | | val1 | | | | | | val2 (optional) | | | | | | | | val3 (optional) | | | | | | | | val4 (optional) | | | | | | | |

**c (2 bits):** A 2-bit, unsigned integer field containing an encoded representation of the number of bytes in this structure.

| Value | Meaning |
| --- | --- |
| ONE\_BYTE\_VAL  0 | Implies that the optional **val2**, **val3**, and **val4** fields are not present. Hence, the structure is 1 byte in size. |
| TWO\_BYTE\_VAL  1 | Implies that the optional **val2** field is present while the optional **val3** and val4 fields are not present. Hence, the structure is 2 bytes in size. |
| THREE\_BYTE\_VAL  2 | Implies that the optional **val2** and **val3** fields are present while the optional **val4** fields are not present. Hence, the structure is 3 bytes in size. |
| FOUR\_BYTE\_VAL  3 | Implies that the optional **val2**, **val3**, and **val4** fields are all present. Hence, the structure is 4 bytes in size. |

**val1 (6 bits):** A 6-bit, unsigned integer field containing the most significant 6 bits of the value represented by this structure.

**val2 (1 byte):** An 8-bit, unsigned integer containing the second most significant bits of the value represented by this structure.

**val3 (1 byte):** An 8-bit, unsigned integer containing the third most significant bits of the value represented by this structure.

**val4 (1 byte):** An 8-bit, unsigned integer containing the least significant bits of the value represented by this structure.

Cache Bitmap - Revision 1 (CACHE\_BITMAP\_ORDER)

The Cache Bitmap - Revision 1 Secondary Drawing Order is used by the server to instruct the client to store a bitmap in a particular Bitmap Cache entry. This order only supports memory-based bitmap caching. Support for the Revision 1 bitmap caches (section [3.1.1.1.1](#Section_2BF9258842BD45278B3EB90C56E292D2)) is specified in the Revision 1 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | cacheId | | | | | | | | pad1Octet | | | | | | | |
| bitmapWidth | | | | | | | | bitmapHeight | | | | | | | | bitmapBitsPerPel | | | | | | | | bitmapLength | | | | | | | |
| ... | | | | | | | | cacheIndex | | | | | | | | | | | | | | | | bitmapComprHdr (optional) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | bitmapDataStream (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** The [Secondary Drawing Order Header (section 2.2.2.2.1.2.1.1)](#Section_c54e2667335d4a59a6e27dc9744dbe79). The embedded **orderType** field MUST be set to one of the following values.

| Value | Meaning |
| --- | --- |
| TS\_CACHE\_BITMAP\_UNCOMPRESSED  0x00 | The bitmap data in the bitmapDataStream field is uncompressed. |
| TS\_CACHE\_BITMAP\_COMPRESSED  0x02 | The bitmap data in the bitmapDataStream field is compressed. |

The embedded **extraFlags** field MAY contain the following flag.

| Value | Meaning |
| --- | --- |
| NO\_BITMAP\_COMPRESSION\_HDR  0x0400 | Indicates that the **bitmapComprHdr** field is not present (removed for bandwidth efficiency to save 8 bytes). |

**cacheId (1 byte):** An 8-bit, unsigned integer. The ID of the bitmap cache in which the bitmap data MUST be stored. This value MUST be in the range 0 to 2 (inclusive).

**pad1Octet (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field are arbitrary and MUST be ignored.

**bitmapWidth (1 byte):** An 8-bit, unsigned integer. The width of the bitmap in pixels.

**bitmapHeight (1 byte):** An 8-bit, unsigned integer. The height of the bitmap in pixels.

**bitmapBitsPerPel (1 byte):** An 8-bit, unsigned integer. The color depth of the bitmap data in bits per pixel. This field MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x08 | 8-bit color depth. |
| 0x10 | 16-bit color depth. |
| 0x18 | 24-bit color depth. |
| 0x20 | 32-bit color depth. |

**bitmapLength (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the data in the **bitmapComprHdr** and **bitmapDataStream** fields.

**cacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the target entry in the destination bitmap cache (specified by the **cacheId** field) where the bitmap data MUST be stored. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the destination bitmap cache. The maximum number of entries allowed in each individual bitmap cache is specified in the Revision 1 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.1) by the **Cache0Entries**, **Cache1Entries**, and **Cache2Entries** fields.

**bitmapComprHdr (8 bytes):** Optional Compressed Data Header structure (see [MS-RDPBCGR] section 2.2.9.1.1.3.1.2.3) describing the bitmap data in the **bitmapDataStream**. This field MUST be present if the TS\_CACHE\_BITMAP\_COMPRESSED (0x02) flag is present in the header field, but the NO\_BITMAP\_COMPRESSION\_HDR (0x0400) flag is not.

**bitmapDataStream (variable):** A variable-length byte array containing bitmap data (the format of this data is defined in [MS-RDPBCGR] section 2.2.9.1.1.3.1.2.2).

Cache Bitmap - Revision 2 (CACHE\_BITMAP\_REV2\_ORDER)

The Cache Bitmap - Revision 2 Secondary Drawing Order is used by the server to instruct the client to store a bitmap in a particular Bitmap Cache entry. This order supports persistent disk bitmap caching and uses a compact encoding format. Support for the Revision 2 bitmap caches (section [3.1.1.1.1](#Section_2BF9258842BD45278B3EB90C56E292D2)) is specified in the Revision 2 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.2).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | key1 (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | key2 (optional) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | bitmapWidth (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bitmapHeight (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bitmapLength (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cacheIndex (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bitmapComprHdr (optional) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bitmapDataStream (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** A [Secondary Drawing Order Header (section 2.2.2.2.1.2.1.1)](#Section_c54e2667335d4a59a6e27dc9744dbe79). The embedded **orderType** field MUST be set to one of the following values.

| Value | Meaning |
| --- | --- |
| TS\_CACHE\_BITMAP\_UNCOMPRESSED\_REV2  0x04 | The bitmap data in the **bitmapDataStream** field is uncompressed. |
| TS\_CACHE\_BITMAP\_COMPRESSED\_REV2  0x05 | The bitmap data in the **bitmapDataStream** field is compressed. |

The format of the embedded **extraFlags** field is specified by the following bitmask diagram.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId | | | bitsPerPixelId | | | | flags | | | | | | | | |

**cacheId (3 bits):** A 3-bit, unsigned integer. The ID of the bitmap cache in which the bitmap data MUST be stored. This value MUST be greater than or equal to 0 and less than the number of bitmap caches being used for the connection. The number of bitmap caches being used is specified by the **NumCellCaches** field of the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2).

**bitsPerPixelId (4 bits):** A 4-bit, unsigned integer. The color depth of the bitmap data in bits per pixel. MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| CBR2\_8BPP  0x3 | 8 bits per pixel |
| CBR2\_16BPP  0x4 | 16 bits per pixel |
| CBR2\_24BPP  0x5 | 24 bits per pixel |
| CBR2\_32BPP  0x6 | 32 bits per pixel |

**flags (9 bits):** A 9-bit, unsigned integer. Operational flags.

| Value | Meaning |
| --- | --- |
| CBR2\_HEIGHT\_SAME\_AS\_WIDTH  0x01 | Implies that the bitmap height is the same as the bitmap width. If this flag is set, the **bitmapHeight** field MUST NOT be present. |
| CBR2\_PERSISTENT\_KEY\_PRESENT  0x02 | Implies that the bitmap is intended to be persisted, and the **key1** and **key2** fields MUST be present. |
| CBR2\_NO\_BITMAP\_COMPRESSION\_HDR  0x08 | Indicates that the **bitmapComprHdr** field is not present (removed for bandwidth efficiency to save 8 bytes). |
| CBR2\_DO\_NOT\_CACHE  0x10 | Implies that the **cacheIndex** field MUST be ignored, and the bitmap MUST be placed in the last entry of the bitmap cache specified by **cacheId** field. |

**key1 (4 bytes):** A 32-bit, unsigned integer. The low 32 bits of the 64-bit persistent bitmap cache key.

**key2 (4 bytes):** A 32-bit, unsigned integer. The high 32 bits of the 64-bit persistent bitmap cache key.

**bitmapWidth (variable):** A [Two-Byte Unsigned Encoding (section 2.2.2.2.1.2.1.2)](#Section_7baeb5f283844d22b6475c66f05be4f7) structure. The width of the bitmap in pixels.

**bitmapHeight (variable):** A Two-Byte Unsigned Encoding (section 2.2.2.2.1.2.1.2) structure. The height of the bitmap in pixels.

**bitmapLength (variable):** A [Four-Byte Unsigned Encoding (section 2.2.2.2.1.2.1.4)](#Section_f6f16e2b876742ee8eff547780cca952) structure. The size in bytes of the data in the **bitmapComprHdr** and **bitmapDataStream** fields.

**cacheIndex (variable):** A Two-Byte Unsigned Encoding (section 2.2.2.2.1.2.1.2) structure. The index of the target entry in the destination bitmap cache (specified by the **cacheId** field) where the bitmap data MUST be stored. If the CBR2\_DO\_NOT\_CACHE flag is not set in the **header** field, the bitmap cache index MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the destination bitmap cache. The maximum number of entries allowed in each individual bitmap cache is specified in the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2) by the **BitmapCache0CellInfo**, **BitmapCache1CellInfo**, **BitmapCache2CellInfo**, **BitmapCache3CellInfo**, and **BitmapCache4CellInfo** fields. If the CBR2\_DO\_NOT\_CACHE flag is set, the **cacheIndex** MUST be set to BITMAPCACHE\_WAITING\_LIST\_INDEX (32767).

**bitmapComprHdr (8 bytes):** Optional Compressed Data Header structure (see [MS-RDPBCGR] section 2.2.9.1.1.3.1.2.3) describing the bitmap data in the **bitmapDataStream**. This field MUST be present if the TS\_CACHE\_BITMAP\_COMPRESSED\_REV2 (0x05) flag is present in the header field, but the CBR2\_NO\_BITMAP\_COMPRESSION\_HDR (0x08) flag is not.

**bitmapDataStream (variable):** A variable-length byte array containing bitmap data (the format of this data is defined in [MS-RDPBCGR] section 2.2.9.1.1.3.1.2.2).

Cache Color Table (CACHE\_COLOR\_TABLE\_ORDER)

The Cache Color Table Secondary Drawing Order is used by the server to instruct the client to store a color table in a particular Color Table Cache entry. Color tables are used in the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders.

Support for color table caching is not specified in the [Color Table Cache Capability Set (section 2.2.1.1)](#Section_2b7c69463612429195a803b7b1387eaf), but is instead implied by support for the MemBlt (section 2.2.2.2.1.1.2.9) and Mem3Blt (section 2.2.2.2.1.1.2.10) Primary Drawing Orders. If support for these orders is advertised in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the existence of a color table cache with entries for six palettes is assumed when palettized color is being used, and the Cache Color Table is used to update these palettes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | cacheIndex | | | | | | | | numberColors | | | | | | | |
| ... | | | | | | | | colorTable (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** A Secondary Order Header, as defined in section [2.2.2.2.1.2.1.1](#Section_c54e2667335d4a59a6e27dc9744dbe79). The embedded **orderType** field MUST be set to TS\_CACHE\_COLOR\_TABLE (0x01).

**cacheIndex (1 byte):** An 8-bit, unsigned integer. An entry in the Cache Color Table where the color table MUST be stored. This value MUST be in the range 0 to 5 (inclusive).

**numberColors (2 bytes):** A 16-bit, unsigned integer. The number of [Color Quad (section 2.2.2.2.1.2.4.1)](#Section_982ce4ebff014eeba5049ba1ee4a722b) structures in the **colorTable** field. This field MUST be set to 256 entries.

**colorTable (variable):** A Color Table composed of an array of Color Quad (section 2.2.2.2.1.2.4.1) structures. The number of entries in the array is given by the **numberColors** field.

Color Quad (TS\_COLOR\_QUAD)

The TS\_COLOR\_QUAD structure is used to express the red, green, and blue components necessary to reproduce a color in the additive RGB space.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| blue | | | | | | | | green | | | | | | | | red | | | | | | | | pad1Octet | | | | | | | |

**blue (1 byte):** An 8-bit, unsigned integer. The blue RGB color component.

**green (1 byte):** An 8-bit, unsigned integer. The green RGB color component.

**red (1 byte):** An 8-bit, unsigned integer. The red RGB color component.

**pad1Octet (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field are arbitrary and MUST be ignored.

Cache Glyph - Revision 1 (CACHE\_GLYPH\_ORDER)

The Cache Glyph - Revision 1 Secondary Drawing Order is used by the server to instruct the client to store a glyph in a particular glyph cache entry (section [3.1.1.1.2](#Section_6D4A8CC0D4814766B609778B09C1CEAC)). Support for glyph caching is specified in the Glyph Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | cacheId | | | | | | | | cGlyphs | | | | | | | |
| glyphData (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| unicodeCharacters (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** A Secondary Order Header, as defined in section [2.2.2.2.1.2.1.1](#Section_c54e2667335d4a59a6e27dc9744dbe79). The embedded **orderType** field MUST be set to TS\_CACHE\_GLYPH (0x03).

The embedded **extraFlags** field MAY contain the following flags.

| Value | Meaning |
| --- | --- |
| CG\_GLYPH\_UNICODE\_PRESENT  0x0100 | Indicates that the **unicodeCharacters** field is present. |

**cacheId (1 byte):** An 8-bit, unsigned integer. The ID of the glyph cache in which the glyph data MUST be stored. This value MUST be in the range 0 to 9 (inclusive).

**cGlyphs (1 byte):** An 8-bit, unsigned integer. The number of glyph entries in the **glyphData** field.

**glyphData (variable):** An array of Cache Glyph Data (section [2.2.2.2.1.2.5.1](#Section_FAF6E70BFD0F4905801061C4439FBCAD)) structures that describes each of the glyphs contained in this order (the number of glyphs is specified by the **cGlyphs** field).

**unicodeCharacters (variable):** An array of Unicode characters. Contains the Unicode character representation of each glyph in the **glyphData** field. The number of bytes in the field is given by **cGlyphs** \* 2. This field MUST NOT be null-terminated. This string is used for diagnostic purposes only and is not necessary for successfully decoding and caching the glyphs in the **glyphData** field.

Cache Glyph Data (TS\_CACHE\_GLYPH\_DATA)

The TS\_CACHE\_GLYPH\_DATA structure contains information describing a single glyph that is to be stored in a glyph cache (section [3.1.1.1.2](#Section_6D4A8CC0D4814766B609778B09C1CEAC)). The ID of this destination glyph cache is specified by the **cacheId** field of the container Cache Glyph (Revision 1) Order (section [2.2.2.2.1.2.5](#Section_6DC06088B1124F4FB0BCB97BBA621A0C)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheIndex | | | | | | | | | | | | | | | | x | | | | | | | | | | | | | | | |
| y | | | | | | | | | | | | | | | | cx | | | | | | | | | | | | | | | |
| cy | | | | | | | | | | | | | | | | aj (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**cacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the target entry in the destination glyph cache where the glyph data MUST be stored. This value MUST be greater than or equal to 0, and less than the maximum number of entries allowed in the destination glyph cache. The maximum number of entries allowed in each of the ten glyph caches is specified in the **GlyphCache** field of the Glyph Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

**x (2 bytes):** A 16-bit, signed integer. The X component of the coordinate that defines the origin of the character within the glyph bitmap. The top-left corner of the bitmap is (0, 0).

**y (2 bytes):** A 16-bit, signed integer. The Y component of the coordinate that defines the origin of the character within the glyph bitmap. The top-left corner of the bitmap is (0, 0).

**cx (2 bytes):** A 16-bit, unsigned integer. The width of the glyph bitmap in pixels.

**cy (2 bytes):** A 16-bit, unsigned integer. The height of the glyph bitmap in pixels.

**aj (variable):** A variable-sized byte array containing a 1-bit-per-pixel bitmap of the glyph. The individual scan lines are encoded in top-down order, and each scan line MUST be byte-aligned. Once the array has been populated with bitmap data, it MUST be padded to a double-word boundary (the size of the structure in bytes MUST be a multiple of 4). For examples of 1-bit-per-pixel encoded glyph bitmaps, see sections [4.6.1](#Section_7f7e6817368146019758c780349f4fcb) and [4.6.2](#Section_095deb9216a64b87bf75f00187b3996a).

Cache Glyph - Revision 2 (CACHE\_GLYPH\_REV2\_ORDER)

The Cache Glyph - Revision 2 Secondary Drawing Order is used by the server to instruct the client to store a glyph in a particular Glyph Cache entry. This order is similar to the Cache Glyph - Revision 1 Secondary Drawing Order (section [2.2.2.2.1.2.5](#Section_6dc06088b1124f4fb0bcb97bba621a0c)) except that it represents glyphs using a more compact format and moves a number of fields into the **extraFlags** field of the secondary order header. Support for glyph caching is specified in the Glyph Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | glyphDataRev2 (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| unicodeCharacters (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** A Secondary Order Header, as defined in section [2.2.2.2.1.2.1.1](#Section_c54e2667335d4a59a6e27dc9744dbe79). The embedded **orderType** field MUST be set to TS\_CACHE\_GLYPH (0x03).

The format of the embedded **extraFlags** word is described by the following bitmask diagram.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId | | | | flags | | | | cGlyphs | | | | | | | |

**cacheId (4 bits):** A 4-bit unsigned integer. The ID of the glyph cache in which the glyph data MUST be stored. This value MUST be in the range 0 to 9 (inclusive).

**flags (4 bits):** A 4-bit, unsigned integer. Various operational flags.

| Value | Meaning |
| --- | --- |
| CG2\_GLYPH\_UNICODE\_PRESENT  0x1 | Indicates that the **unicodeCharacters** field is present. |
| GLYPH\_ORDER\_REV2  0x2 | Indicates that this is a Cache Glyph - Revision 2 Order; the Cache Glyph - Revision 1 (section 2.2.2.2.1.2.5) **extraFlags** header field does not contain a flag with this value. |

**cGlyphs (1 byte):** An 8-bit, unsigned integer. The number of glyph entries in the **glyphData** field.

**glyphDataRev2 (variable):** The specification for each of the glyphs in this order (the number of glyphs is given by the **cGlyphs** field embedded in the **header** field) defined using Cache Glyph Data - Revision 2 (section [2.2.2.2.1.2.6.1)](#Section_7facfe557db14a47a38ebc223cce5dc4) structures.

**unicodeCharacters (variable):** An array of Unicode characters. Contains the Unicode character representation of each glyph in the **glyphData** field. The number of bytes in the field is given by **cGlyphs** \* 2 (where **cGlyphs** is embedded in the **header** field). This field MUST NOT be null-terminated. This string is used for diagnostic purposes only and is not necessary for successfully decoding and caching the glyphs in the **glyphData** field.

Cache Glyph Data - Revision 2 (TS\_CACHE\_GLYPH\_DATA\_REV2)

The TS\_CACHE\_GLYPH\_DATA\_REV2 structure contains information describing a single glyph that is to be stored in a glyph cache (section [3.1.1.1.2](#Section_6D4A8CC0D4814766B609778B09C1CEAC)). The ID of this destination glyph cache is specified by the **cacheId** field of the container Cache Glyph (Revision 2) Order (section [2.2.2.2.1.2.6](#Section_FFC80435136E442E908540CA1D27A413)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheIndex | | | | | | | | x (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| y (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cx (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cy (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| aj (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**cacheIndex (1 byte):** An 8-bit, unsigned integer. The index of the target entry in the destination glyph cache where the glyph data MUST be stored. This value MUST be greater than or equal to 0, and less than the maximum number of entries allowed in the destination glyph cache. The maximum number of entries allowed in each of the ten glyph caches is specified in the **GlyphCache** field of the Glyph Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

**x (variable):** A [Two-Byte Signed Encoding (section 2.2.2.2.1.2.1.3)](#Section_7dcab5880143463fa11c63bbd6f30090) structure. The X component of the coordinate that defines the origin of the character within the glyph bitmap. The top-left corner of the bitmap is (0, 0).

**y (variable):** A Two-Byte Signed Encoding (section 2.2.2.2.1.2.1.3) structure. The Y component of the coordinate that defines the origin of the character within the glyph bitmap. The top-left corner of the bitmap is (0, 0).

**cx (variable):** A [Two-Byte Unsigned Encoding (section 2.2.2.2.1.2.1.2)](#Section_7baeb5f283844d22b6475c66f05be4f7) structure. The width of the glyph bitmap in pixels.

**cy (variable):** A Two-Byte Unsigned Encoding (section 2.2.2.2.1.2.1.2) structure. The height of the glyph bitmap in pixels.

**aj (variable):** A variable-sized byte array containing a 1-bit-per-pixel bitmap of the glyph.

The individual scan lines are encoded in top-down order, and each scan line MUST be byte-aligned. After the array has been populated with bitmap data, it MUST be padded to a double-word boundary (the size of the structure in bytes MUST be a multiple of 4). The size, in bytes, of the glyph data is given by the following function.

1. ROUND\_UP\_TO\_NEAREST\_MULTIPLE\_OF\_4(((cx + 7) / 8) \* cy)

For examples of 1-bit-per-pixel encoded glyph bitmaps, see sections [4.6.1](#Section_7f7e6817368146019758c780349f4fcb) and [4.6.2](#Section_095deb9216a64b87bf75f00187b3996a).

Cache Brush (CACHE\_BRUSH\_ORDER)

The Cache Brush Secondary Drawing Order is used by the server to instruct the client to store a brush in a particular Brush Cache entry. Support for brush caching is specified in the Brush Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | cacheEntry | | | | | | | | iBitmapFormat | | | | | | | |
| cx | | | | | | | | cy | | | | | | | | Style | | | | | | | | iBytes | | | | | | | |
| brushData (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** Secondary Order Header, as defined in section [2.2.2.2.1.2.1.1](#Section_c54e2667335d4a59a6e27dc9744dbe79). The embedded **orderType** field MUST be set to TS\_CACHE\_BRUSH (0x07).

**cacheEntry (1 byte):** An 8-bit, unsigned integer. The entry in a specified Brush Cache where the brush data MUST be stored. This value MUST be in the range 0 to 63 (inclusive).

**iBitmapFormat (1 byte):** An 8-bit, unsigned integer. The color depth of the brush bitmap data. This field MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| BMF\_1BPP  0x01 | 1 bit per pixel |
| BMF\_8BPP  0x03 | 8 bits per pixel |
| BMF\_16BPP  0x04 | 15 or 16 bits per pixel |
| BMF\_24BPP  0x05 | 24 bits per pixel |
| BMF\_32BPP  0x06 | 32 bits per pixel |

**cx (1 byte):** An 8-bit, unsigned integer. The width of the brush bitmap.

**cy (1 byte):** An 8-bit, unsigned integer. The height of the brush bitmap.

**Style (1 byte):** An 8-bit, unsigned integer. This field is not used, and SHOULD[<4>](#Appendix_A_4) be set to 0x00.

**iBytes (1 byte):** An 8-bit, unsigned integer. The size of the **brushData** field in bytes.

**brushData (variable):** A variable-sized byte array containing binary brush data that represents an 8-by-8-pixel bitmap image. There are 64 pixels in a brush bitmap, and the space used to represent each pixel depends on the color depth of the brush bitmap and the number of colors used. The size of the **brushData** field in bytes is given by the **iBytes** field.

In general, most brushes only use two colors (mono format), and the majority of the remaining ones use four colors or fewer.

For mono format brushes (**iBitmapFormat** is BMF\_1BPP), **brushData** contains 8 bytes of 1-bit-per-pixel data, each byte corresponding to a row of pixels in the brush. The rows are encoded in reverse order; that is, the pixels in the bottom row of the brush are encoded in the first byte of the **brushData** field, and the pixels in the top row are encoded in the eighth byte.

For color brushes, a compression algorithm is used. If the data is compressed, the **iBytes** field is 20 for 256-color (**iBitmapFormat** is BMF\_8BPP), 24 for 16-bit color (**iBitmapFormat** is BMF\_16BPP), 28 for 24-bit color (**iBitmapFormat** is BMF\_24BPP), and 32 for 32-bit color (**iBitmapFormat** is BMF\_32BPP). The compression algorithm reduces brush data size by storing each brush pixel as a 2-bit index (four possible values) into a translation table containing four entries. This equates to 2 bytes per brush bitmap line (16 bytes in total) followed by the translation table contents. This layout for four-color brushes conforms to the [Compressed Color Brush (section 2.2.2.2.1.2.7.1)](#Section_43f2b0a56a7647a3b6d279b08b80f2d2) structure.

For brushes using more than four colors, the data is simply copied uncompressed into the **brushData** at the appropriate color depth.

Compressed Color Brush (COMPRESSED\_COLOR\_BRUSH)

The COMPRESSED\_COLOR\_BRUSH structure is used to hold a compressed version of a four-color 8-by-8-pixel brush.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| P1 | | P2 | | P3 | | P4 | | P5 | | P6 | | P7 | | P8 | | P9 | | P10 | | P11 | | P12 | | P13 | | P14 | | P15 | | P16 | |
| P17 | | P18 | | P19 | | P20 | | P21 | | P22 | | P23 | | P24 | | P25 | | P26 | | P27 | | P28 | | P29 | | P30 | | P31 | | P32 | |
| P33 | | P34 | | P35 | | P36 | | P37 | | P38 | | P39 | | P40 | | P41 | | P42 | | P43 | | P44 | | P45 | | P46 | | P47 | | P48 | |
| P49 | | P50 | | P51 | | P52 | | P53 | | P54 | | P55 | | P56 | | P57 | | P58 | | P59 | | P60 | | P61 | | P62 | | P63 | | P64 | |
| color1 (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| color2 (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| color3 (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| color4 (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**P1 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the first pixel.

| Value (2 bits) | Meaning |
| --- | --- |
| 0 | Use the first value in the translation table (**color1**). |
| 1 | Use the second value in the translation table (**color2**). |
| 2 | Use the third value in the translation table (**color3**). |
| 3 | Use the fourth value in the translation table (**color4**). |

**P2 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the second pixel. The possible values for this field are identical to those for P1.

**P3 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the third pixel. The possible values for this field are identical to those for P1.

**P4 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the fourth pixel. The possible values for this field are identical to those for P1.

**P5 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the fifth pixel. The possible values for this field are identical to those for P1.

**P6 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the sixth pixel. The possible values for this field are identical to those for P1.

**P7 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the seventh pixel. The possible values for this field are identical to those for P1.

**P8 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the eighth pixel. The possible values for this field are identical to those for P1.

**P9 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the ninth pixel. The possible values for this field are identical to those for P1.

**P10 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the tenth pixel. The possible values for this field are identical to those for P1.

**P11 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 11th pixel. The possible values for this field are identical to those for P1.

**P12 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 12th pixel. The possible values for this field are identical to those for P1.

**P13 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 13th pixel. The possible values for this field are identical to those for P1.

**P14 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 14th pixel. The possible values for this field are identical to those for P1.

**P15 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 15th pixel. The possible values for this field are identical to those for P1.

**P16 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 16th pixel. The possible values for this field are identical to those for P1.

**P17 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 17th pixel. The possible values for this field are identical to those for P1.

**P18 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 18th pixel. The possible values for this field are identical to those for P1.

**P19 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 19th pixel. The possible values for this field are identical to those for P1.

**P20 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 20th pixel. The possible values for this field are identical to those for P1.

**P21 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 21st pixel. The possible values for this field are identical to those for P1.

**P22 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 22nd pixel. The possible values for this field are identical to those for P1.

**P23 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 23rd pixel. The possible values for this field are identical to those for P1.

**P24 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 24th pixel. The possible values for this field are identical to those for P1.

**P25 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 25th pixel. The possible values for this field are identical to those for P1.

**P26 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 26th pixel. The possible values for this field are identical to those for P1.

**P27 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 27th pixel. The possible values for this field are identical to those for P1.

**P28 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 28th pixel. The possible values for this field are identical to those for P1.

**P29 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 29th pixel. The possible values for this field are identical to those for P1.

**P30 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 30th pixel. The possible values for this field are identical to those for P1.

**P31 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 31st pixel. The possible values for this field are identical to those for P1.

**P32 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 32nd pixel. The possible values for this field are identical to those for P1.

**P33 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 33rd pixel. The possible values for this field are identical to those for P1.

**P34 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 34th pixel. The possible values for this field are identical to those for P1.

**P35 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 35th pixel. The possible values for this field are identical to those for P1.

**P36 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 36th pixel. The possible values for this field are identical to those for P1.

**P37 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 37th pixel. The possible values for this field are identical to those for P1.

**P38 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 38th pixel. The possible values for this field are identical to those for P1.

**P39 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 39th pixel. The possible values for this field are identical to those for P1.

**P40 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 40th pixel. The possible values for this field are identical to those for P1.

**P41 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 41st pixel. The possible values for this field are identical to those for P1.

**P42 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 42nd pixel. The possible values for this field are identical to those for P1.

**P43 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 43rd pixel. The possible values for this field are identical to those for P1.

**P44 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 44th pixel. The possible values for this field are identical to those for P1.

**P45 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 45th pixel. The possible values for this field are identical to those for P1.

**P46 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 46th pixel. The possible values for this field are identical to those for P1.

**P47 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 47th pixel. The possible values for this field are identical to those for P1.

**P48 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 48th pixel. The possible values for this field are identical to those for P1.

**P49 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 49th pixel. The possible values for this field are identical to those for P1.

**P50 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 50th pixel. The possible values for this field are identical to those for P1.

**P51 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 51st pixel. The possible values for this field are identical to those for P1.

**P52 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 52nd pixel. The possible values for this field are identical to those for P1.

**P53 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 53rd pixel. The possible values for this field are identical to those for P1.

**P54 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 54th pixel. The possible values for this field are identical to those for P1.

**P55 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 55th pixel. The possible values for this field are identical to those for P1.

**P56 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 56th pixel. The possible values for this field are identical to those for P1.

**P57 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 57th pixel. The possible values for this field are identical to those for P1.

**P58 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 58th pixel. The possible values for this field are identical to those for P1.

**P59 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 59th pixel. The possible values for this field are identical to those for P1.

**P60 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 60th pixel. The possible values for this field are identical to those for P1.

**P61 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 61st pixel. The possible values for this field are identical to those for P1.

**P62 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 62nd pixel. The possible values for this field are identical to those for P1.

**P63 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 63rd pixel. The possible values for this field are identical to those for P1.

**P64 (2 bits):** A 2-bit, unsigned integer field. A 2-bit value indicating the translation table entry to use for the 64th pixel. The possible values for this field are identical to those for P1.

**color1 (variable):** Translation table entry 1. This entry is an index into the current color palette or an RGB triplet value; the actual interpretation depends on the color depth of the bitmap data.

| Color depth | Field size | Meaning |
| --- | --- | --- |
| 8 bpp | 1 byte | Index into the current color palette. |
| 15 bpp | 2 bytes | RGB color triplet expressed in 5-5-5 format (5 bits for red, 5 bits for green, and 5 bits for blue). |
| 16 bpp | 2 bytes | RGB color triplet expressed in 5-6-5 format (5 bits for red, 6 bits for green, and 5 bits for blue). |
| 24 bpp | 3 bytes | RGB color triplet (1 byte per component). |

**color2 (variable):** Translation table entry 2. This entry is an index into the current color palette or an RGB triplet value; the actual interpretation depends on the color depth of the bitmap data. The possible values for this field are identical to those for **color1**.

**color3 (variable):** Translation table entry 3. This entry is an index into the current color palette or an RGB triplet value; the actual interpretation depends on the color depth of the bitmap data. The possible values for this field are identical to those for **color1**.

**color4 (variable):** Translation table entry 4. This entry is an index into the current color palette or an RGB triplet value; the actual interpretation depends on the color depth of the bitmap data. The possible values for this field are identical to those for **color1**.

Cache Bitmap - Revision 3 (CACHE\_BITMAP\_REV3\_ORDER)

The Cache Bitmap - Revision 3 Secondary Drawing Order is used by the server to instruct the client to store a bitmap in a particular Bitmap Cache entry. This order supports persistent disk bitmap caching and also enables the use of the bitmap codecs specified in the Bitmap Codec Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.2.10). Support for the Cache Bitmap - Revision 3 Secondary Drawing Order is specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | cacheIndex | | | | | | | | | | | | | | | |
| key1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| key2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bitmapData (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (6 bytes):** A Secondary Drawing Order Header (see section [2.2.2.2.1.2.1.1](#Section_c54e2667335d4a59a6e27dc9744dbe79)). The embedded **orderType** field MUST be set to TS\_CACHE\_BITMAP\_COMPRESSED\_REV3 (8).

The format of the embedded **extraFlags** field is specified by the following bitmask diagram.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cacheId | | | bitsPerPixelId | | | | flags | | | | | | | | |

**cacheId (3 bits):** A 3-bit, unsigned integer. The ID of the bitmap cache in which bitmap data MUST be stored. This value MUST be greater than or equal to 0 and less than the number of bitmap caches being used for the connection. The number of bitmap caches being used is specified by the **NumCellCaches** field of the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2).

**bitsPerPixelId (4 bits):** A 4-bit, unsigned integer. The color depth of the bitmap data in bits per pixel. It MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| CBR23\_8BPP  0x3 | 8 bits per pixel |
| CBR23\_16BPP  0x4 | 16 bits per pixel |
| CBR23\_24BPP  0x5 | 24 bits per pixel |
| CBR23\_32BPP  0x6 | 32 bits per pixel |

**flags (9 bits):** A 9-bit, unsigned integer. Operational flags.

| Value | Meaning |
| --- | --- |
| CBR3\_IGNORABLE\_FLAG  0x08 | This flag has no meaning and its value is ignored by the client. |
| CBR3\_DO\_NOT\_CACHE  0x10 | Implies that the cacheIndex field MUST be ignored, and the bitmap MUST be placed in the last entry of the bitmap cache specified by the cacheId field. |

**cacheIndex (2 bytes):** A 16-bit unsigned integer. The index of the target entry in the destination bitmap cache (specified by the **cacheId** field) where the bitmap data MUST be stored. If the CBR3\_DO\_NOT\_CACHE flag is not set in the **header** field, the bitmap cache index MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the destination bitmap cache. The maximum number of entries allowed in each individual bitmap cache is specified in the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2) by the **BitmapCache0CellInfo**, **BitmapCache1CellInfo**, **BitmapCache2CellInfo**, **BitmapCache3CellInfo**, and **BitmapCache4CellInfo** fields. If the CBR3\_DO\_NOT\_CACHE flag is set, the **cacheIndex** MUST be set to BITMAPCACHE\_WAITING\_LIST\_INDEX (32767).

**key1 (4 bytes):** A 32-bit, unsigned integer. The low 32 bits of the 64-bit persistent bitmap cache key.

**key2 (4 bytes):** A 32-bit, unsigned integer. The high 32 bits of the 64-bit persistent bitmap cache key.

**bitmapData (variable):** An Extended Bitmap Data (see [MS-RDPBCGR] section 2.2.9.2.1.1) structure that contains an encoded bitmap image.

###### Alternate Secondary Drawing Orders

Common Data Types

Alternate Secondary Drawing Order Header (ALTSEC\_DRAWING\_ORDER\_HEADER)

The ALTSEC\_DRAWING\_ORDER\_HEADER structure is included in all alternate secondary drawing orders.

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

**controlFlags (1 byte):** An 8-bit, unsigned integer. The control byte that identifies the class and type of the drawing order.

The format of the **controlFlags** byte is described by the following bitmask diagram.

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

**class (2 bits):** A 2-bit, unsigned integer. This field MUST contain only the TS\_SECONDARY (0x02) flag to indicate that the order is an alternate secondary drawing order (see section [2.2.2.2.1](#Section_5a67b33f446d41059355fe5de076975c)).

**orderType (6 bits):** A 6-bit, unsigned integer. Identifies the type of alternate secondary drawing order.

| Value | Meaning |
| --- | --- |
| TS\_ALTSEC\_SWITCH\_SURFACE  0x00 | Switch Surface Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.3](#Section_ada4d8a9421e48c0a0ed7384fa3cf061)). |
| TS\_ALTSEC\_CREATE\_OFFSCR\_BITMAP  0x01 | Create Offscreen Bitmap Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.2](#Section_427758ff4bac4833b388d634cc512d0c)). |
| TS\_ALTSEC\_STREAM\_BITMAP\_FIRST  0x02 | Stream Bitmap First (Revision 1 and 2) Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.5.1](#Section_b714a7c97fd54621a397349a4ef3e06c)). |
| TS\_ALTSEC\_STREAM\_BITMAP\_NEXT  0x03 | Stream Bitmap Next Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.5.2](#Section_dfe099ffa894406393a7fbe64e7cb85e)). |
| TS\_ALTSEC\_CREATE\_NINEGRID\_BITMAP  0x04 | Create NineGrid Bitmap Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.4](#Section_6ac5eb820c30405490e883797f24bc38)). |
| TS\_ALTSEC\_GDIP\_FIRST  0x05 | Draw GDI+ First Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.6.2](#Section_53dbc9c54e3f4b17a256baf43c582146)). |
| TS\_ALTSEC\_GDIP\_NEXT  0x06 | Draw GDI+ Next Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.6.3](#Section_20da0b5e8cc94805998e03e9f216bade)). |
| TS\_ALTSEC\_GDIP\_END  0x07 | Draw GDI+ End Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.6.4](#Section_b44bd2b06c00454db182dbb841131c7a)). |
| TS\_ALTSEC\_GDIP\_CACHE\_FIRST  0x08 | Draw GDI+ First Alternate Secondary Drawing Order (see section 2.2.2.2.1.3.6.2). |
| TS\_ALTSEC\_GDIP\_CACHE\_NEXT  0x09 | Draw GDI+ Cache Next Alternate Secondary Drawing Order (see section 2.2.2.2.1.3.6.3). |
| TS\_ALTSEC\_GDIP\_CACHE\_END  0x0A | Draw GDI+ Cache End Alternate Secondary Drawing Order (see section 2.2.2.2.1.3.6.4). |
| TS\_ALTSEC\_WINDOW  0x0B | Windowing Alternate Secondary Drawing Order (see [[MS-RDPERP]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPERP%5d.pdf) section 2.2.1.3). |
| TS\_ALTSEC\_COMPDESK\_FIRST  0x0C | Desktop Composition Alternate Secondary Drawing Order (see [[MS-RDPEDC]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPEDC%5d.pdf) section 2.2.1.1). |
| TS\_ALTSEC\_FRAME\_MARKER  0x0D | Frame Marker Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.7](#Section_9ba0e3d204ae4c73a35ecb854f587018)). |

Create Offscreen Bitmap (CREATE\_OFFSCR\_BITMAP\_ORDER)

The Create Offscreen Bitmap Alternate Secondary Drawing Order is used by the server to instruct the client to create a bitmap of a particular width and height in the Offscreen Bitmap Cache. Support for offscreen bitmap caching is specified in the Offscreen Bitmap Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | flags | | | | | | | | | | | | | | | | cx | | | | | | | |
| ... | | | | | | | | cy | | | | | | | | | | | | | | | | deleteList (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded orderType field MUST be set to TS\_ALTSEC\_CREATE\_OFFSCR\_BITMAP (0x01).

**flags (2 bytes):** A 16-bit, unsigned integer. Operational flags. The format of the flags field is described by the following bitmask diagram:

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

**offscreenBitmapId (15 bits):** A 15-bit unsigned integer. The index of the Offscreen Bitmap Cache entry wherein the bitmap MUST be created. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the Offscreen Bitmap Cache as specified by the **offscreenCacheEntries** field of the Offscreen Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.9).

**d (1 bit):** A 1-bit field. Indicates that the **deleteList** field is present.

**cx (2 bytes):** A 16-bit, unsigned integer. The width in pixels of the offscreen bitmap to create.

**cy (2 bytes):** A 16-bit, unsigned integer. The height in pixels of the offscreen bitmap to create.

**deleteList (variable):** A collection of Offscreen Bitmap Cache entries that MUST be deleted, stored in an [Offscreen Cache Delete List (section 2.2.2.2.1.3.2.1)](#Section_e40b1297f5ad46c8a83ee7529f1d5c5b) structure.

Offscreen Cache Delete List (OFFSCR\_DELETE\_LIST)

The OFFSCR\_DELETE\_LIST structure is used to encode a collection of Offscreen Bitmap Cache indices that MUST be deleted.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cIndices | | | | | | | | | | | | | | | | indices (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**cIndices (2 bytes):** A 16-bit, unsigned integer. The number of 2-byte indices held in the **indices** field.

**indices (variable):** A collection of offscreen bitmap cache indices that identify cache entries that MUST be deleted. Each index is a 16-bit unsigned integer and MUST be greater than or equal to 0, and less than the maximum number of entries allowed in the Offscreen Bitmap Cache, as specified by the **offscreenCacheEntries** field of the Offscreen Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9). The number of indices in this list is specified by the **cIndices** field.

Switch Surface (SWITCH\_SURFACE\_ORDER)

The Switch Surface Alternate Secondary Drawing Order is used by the server to instruct the client to switch the target drawing surface either to the primary drawing surface (screen desktop) or to an entry in the Offscreen Bitmap Cache. Support for offscreen bitmap caching is specified in the Offscreen Bitmap Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9).

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

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_SWITCH\_SURFACE (0x00).

**bitmapId (2 bytes):** A 16-bit, unsigned integer. The new target drawing surface. If this field has a value less than SCREEN\_BITMAP\_SURFACE (0xFFFF), it identifies an entry in the Offscreen Bitmap Cache which contains a bitmap surface that MUST become the new target drawing surface. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the Offscreen Bitmap Cache as specified by the **offscreenCacheEntries** field of the Offscreen Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.9). Otherwise, if this field has the value SCREEN\_BITMAP\_SURFACE, the target drawing surface MUST be changed to the primary drawing surface (screen desktop).

Create NineGrid Bitmap (CREATE\_NINEGRID\_BITMAP\_ORDER)

The Create NineGrid Bitmap Alternate Secondary Drawing Order is used by the server to instruct the client to create a NineGrid bitmap of a particular width and height in the NineGrid Bitmap Cache (the color depth MUST be 32 bpp). Support for NineGrid drawing is specified in the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | BitmapBpp | | | | | | | | BitmapId | | | | | | | | | | | | | | | |
| cx | | | | | | | | | | | | | | | | cy | | | | | | | | | | | | | | | |
| nineGridInfo (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_CREATE\_NINEGRID\_BITMAP (0x04).

**BitmapBpp (1 byte):** An 8-bit, unsigned integer. The color depth in bits per pixel of the NineGrid bitmap to create. Currently, all NineGrid bitmaps are sent in 32 bpp, so this field MUST be set to 0x20.

**BitmapId (2 bytes):** A 16-bit, unsigned integer. The index of the NineGrid Bitmap Cache entry wherein the bitmap and NineGrid transformation information MUST be stored. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the NineGrid Bitmap Cache as specified by the **drawNineGridCacheEntries** field of the DrawNineGrid Cache Capability Set (section 2.2.1.2).

**cx (2 bytes):** A 16-bit, unsigned integer. The width in pixels of the NineGrid bitmap to create.

**cy (2 bytes):** A 16-bit, unsigned integer. The height in pixels of the NineGrid bitmap to create.

**nineGridInfo (16 bytes):** A [NineGrid Bitmap Information (section 2.2.2.2.1.3.4.1)](#Section_ba44b18260ad4bb199d55d00c0695bcb) structure that describes properties of the NineGrid bitmap to be created.

NineGrid Bitmap Information (NINEGRID\_BITMAP\_INFO)

The NINEGRID\_BITMAP\_INFO structure is used to describe a NineGrid source bitmap (see section [4.4](#Section_06ba35beca674fd1937e7a0ac678b63a)). For more information about NineGrid bitmaps, see [[NINEGRID]](http://go.microsoft.com/fwlink/?LinkId=90233).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| flFlags | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ulLeftWidth | | | | | | | | | | | | | | | | ulRightWidth | | | | | | | | | | | | | | | |
| ulTopHeight | | | | | | | | | | | | | | | | ulBottomHeight | | | | | | | | | | | | | | | |
| crTransparent | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**flFlags (4 bytes):** A 32-bit, unsigned integer. Option flags for the NineGrid bitmap represented by this structure.

| Value | Meaning |
| --- | --- |
| DSDNG\_STRETCH  0x00000001 | Indicates that the center portion of the source bitmap MUST be stretched to fill the center of the destination NineGrid. |
| DSDNG\_TILE  0x00000002 | Indicates that the center portion source bitmap MUST be tiled to fill the center of the destination NineGrid. |
| DSDNG\_PERPIXELALPHA  0x00000004 | Indicates that an AlphaBlend operation MUST be used to compose the destination NineGrid. The source bitmap is expected to have per-pixel alpha values. For a description of the AlphaBlend operation, see [[MSDN-ABLEND]](http://go.microsoft.com/fwlink/?LinkId=89948). |
| DSDNG\_TRANSPARENT  0x00000008 | Indicates that a TransparentBlt operation MUST be used to compose the destination NineGrid. The **crTransparent** field MUST contain the transparent color. For a description of the TransparentBlt operation, see [[MSDN-TransparentBlt]](http://go.microsoft.com/fwlink/?LinkId=90146). |
| DSDNG\_MUSTFLIP  0x00000010 | Indicates that the source NineGrid MUST be flipped on a vertical axis. |
| DSDNG\_TRUESIZE  0x00000020 | Indicates that the source bitmap MUST be transferred without stretching or tiling. |

If the DSDNG\_TILE (0x00000002) flag is not set, the DSDNG\_STRETCH (0x00000001) flag is implied. If neither the DSDNG\_PERPIXELALPHA (0x00000004) nor DSDNG\_TRANSPARENT (0x00000008) is indicated, a BitBlt operation MUST be applied. For a description of the BitBlt operation, see [[MSDN-BitBlt]](http://go.microsoft.com/fwlink/?LinkId=89958).

**ulLeftWidth (2 bytes):** A 16-bit, unsigned integer. The width of the left-side NineGrid border. For a visual illustration of this field, see section 4.4.

**ulRightWidth (2 bytes):** A 16-bit, unsigned integer. The width of the right-side NineGrid border. For a visual illustration of this field, see section 4.4.

**ulTopHeight (2 bytes):** A 16-bit, unsigned integer. The height of the top NineGrid border. For a visual illustration of this field, see section 4.4.

**ulBottomHeight (2 bytes):** A 16-bit, unsigned integer. The height of the bottom NineGrid border. For a visual illustration of this field, see section 4.4.

**crTransparent (4 bytes):** The RGB color in the source bitmap to treat as transparent represented using a [Color Reference (section 2.2.2.2.1.3.4.1.1)](#Section_d8a54901807c4b3fb7c9d50cf01b62ce) structure. This field is used if the DSDNG\_TRANSPARENT (0x00000008) flag is set in the **flFlags** field.

Color Reference (TS\_COLORREF)

The TS\_COLORREF structure is used to express the red, green, and blue components necessary to reproduce a color in the additive RGB space.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| red | | | | | | | | green | | | | | | | | blue | | | | | | | | zeroPad | | | | | | | |

**red (1 byte):** An 8-bit, unsigned integer. The red RGB color component.

**green (1 byte):** An 8-bit, unsigned integer. The green RGB color component.

**blue (1 byte):** An 8-bit, unsigned integer. The blue RGB color component.

**zeroPad (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field MUST be ignored. This field MUST be set to zero.

Stream Bitmap Orders

The stream bitmap alternate secondary orders (added in RDP version 5.1) enable bitmaps of arbitrary size to be broken up into 4,096-byte blocks and streamed from server to client, each block being sent in a separate stream bitmap order. All of the bitmap blocks MUST be sent in sequence. The bitmaps can be sent either compressed or uncompressed.

None of the stream bitmap alternate secondary orders include the cache entry in which to store the streamed bitmap. This is because the server MUST always send a bitmap creation order that contains the bitmap cache entry data immediately after streaming the bitmap bits to the client.

If support for NineGrid rendering is specified using the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780), the support for bitmap streaming is implicitly assumed to be the case, as the NineGrid bitmaps are transported using bitmap streaming.

Stream Bitmap First (STREAM\_BITMAP\_FIRST\_ORDER)

The Stream Bitmap First Alternate Secondary Drawing Order is used by the server to send the client the first block in a streamed bitmap and information describing the bitmap (such as color depth, width, and height).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | BitmapFlags | | | | | | | | BitmapBpp | | | | | | | | BitmapType | | | | | | | |
| ... | | | | | | | | BitmapWidth | | | | | | | | | | | | | | | | BitmapHeight | | | | | | | |
| ... | | | | | | | | BitmapSize (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BitmapBlockSize | | | | | | | | | | | | | | | | BitmapBlock (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as specified in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_STREAM\_BITMAP\_FIRST (0x02).

**BitmapFlags (1 byte):** An 8-bit, unsigned integer. Flags describing the order contents and layout.

| Value | Meaning |
| --- | --- |
| STREAM\_BITMAP\_END  0x01 | Indicates that the bitmap fits into one stream bitmap block (4,096 bytes). |
| STREAM\_BITMAP\_COMPRESSED  0x02 | Indicates that the bitmap data is compressed. |
| STREAM\_BITMAP\_REV2  0x04 | Indicates that the BitmapSize field is 4 bytes. If this flag is not set, the **BitmapSize** field is 2 bytes. |

**BitmapBpp (1 byte):** An 8-bit, unsigned integer. The color depth in bits per pixel of the streamed bitmap.

**BitmapType (2 bytes):** A 16-bit, unsigned integer. The type of the streamed bitmap.

| Value | Meaning |
| --- | --- |
| TS\_DRAW\_NINEGRID\_BITMAP\_CACHE  0x0001 | Indicates that the data in the **BitmapBlock** field is a NineGrid source bitmap. |

**BitmapWidth (2 bytes):** A 16-bit, unsigned integer. The width in pixels of the streamed bitmap.

**BitmapHeight (2 bytes):** A 16-bit, unsigned integer. The height in pixels of the streamed bitmap.

**BitmapSize (variable):** A variable-length field containing the total size in bytes of the streamed bitmap. If the STREAM\_BITMAP\_REV2 (0x04) flag is set in the **BitmapFlags** field, this field MUST contain a 32-bit unsigned integer. If the STREAM\_BITMAP\_REV2 flag is not set, this field MUST contain a 16-bit unsigned integer.

**BitmapBlockSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the bitmap stream data block contained in the **BitmapBlock** field. This value MUST be less than or equal to the value contained in the **BitmapSize** field; if the STREAM\_BITMAP\_END (0x01) flag is set in the **BitmapFlags** field, the two values MUST be equal.

**BitmapBlock (variable):** A variable-length byte array. The first block of the streamed bitmap (also the last if the STREAM\_BITMAP\_END (0x01) flag is set in the **BitmapFlags** field). The size of this block is given by the **BitmapBlockSize** field.

Stream Bitmap Next (STREAM\_BITMAP\_NEXT\_ORDER)

The Stream Bitmap Next Alternate Secondary Drawing Order is used by the server to send the client intermediate and final blocks in a streamed bitmap.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | BitmapFlags | | | | | | | | BitmapType | | | | | | | | | | | | | | | |
| BitmapBlockSize | | | | | | | | | | | | | | | | BitmapBlock (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_STREAM\_BITMAP\_NEXT (0x03).

**BitmapFlags (1 byte):** An 8-bit, unsigned integer. Flags describing the order contents and layout.

| Value | Meaning |
| --- | --- |
| STREAM\_BITMAP\_END  0x01 | Indicates that this packet contains the final stream bitmap block. |
| STREAM\_BITMAP\_COMPRESSED  0x02 | Indicates that the bitmap data is compressed. |

**BitmapType (2 bytes):** A 16-bit, unsigned integer. The type of the streamed bitmap.

| Value | Meaning |
| --- | --- |
| TS\_DRAW\_NINEGRID\_BITMAP\_CACHE  0x0001 | Indicates that the data in the BitmapBlock field is a NineGrid source bitmap. |

**BitmapBlockSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the bitmap stream data block contained in the **BitmapBlock** field.

**BitmapBlock (variable):** A variable-length byte array. The intermediate or final block of the streamed bitmap. The size of this block is given by the **BitmapBlockSize** field.

GDI+ Orders

Common Data Types

GDI+ Cache Type (DRAW\_GDIPLUS\_CACHE\_TYPE)

The DRAW\_GDIPLUS\_CACHE\_TYPE structure stores a GDI+ cache type identifier. The Cache Type is derived from the Type field of an EMF+ Record (see [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3). Only a subset of these types are cacheable.

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

**cacheType (2 bytes):** A 16-bit, unsigned integer. The GDI+ cache type MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| GDIP\_CACHE\_GRAPHICS\_DATA  0x0001 | GDI+ Graphics Cache.  This cache type is associated with EMF+ Records of type EmfPlusSetTSGraphics (see [MS-EMFPLUS] section 2.3.8.2). |
| GDIP\_CACHE\_OBJECT\_BRUSH  0x0002 | GDI+ Brush Cache.  This cache type is associated with EMF+ Records of type EmfPlusObject (see [MS-EMFPLUS] section 2.3.5.1) where the object type is ObjectTypeBrush (see [MS-EMFPLUS] section 2.1.1.22). |
| GDIP\_CACHE\_OBJECT\_PEN  0x0003 | GDI+ Pen Cache.  This cache type is associated with EMF+ Records of type EmfPlusObject (see [MS-EMFPLUS] section 2.3.5.1) where the object type is ObjectTypePen (see [MS-EMFPLUS] section 2.1.1.22). |
| GDIP\_CACHE\_OBJECT\_IMAGE  0x0004 | GDI+ Image Cache.  This cache type is associated with EMF+ Records of type EmfPlusObject (see [MS-EMFPLUS] section 2.3.5.1) where the object type is ObjectTypeImage (see [MS-EMFPLUS] section 2.1.1.22). |
| GDIP\_CACHE\_OBJECT\_IMAGEATTRIBUTES  0x0005 | GDI+ Image Attributes Cache.  This cache type is associated with EMF+ Records of type EmfPlusObject (see [MS-EMFPLUS] section 2.3.5.1) where the object type is ObjectTypeImageAttributes (see [MS-EMFPLUS] section 2.1.1.22). |

Draw GDI+ Cache First (DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER)

The Draw GDI+ Cache First Alternate Secondary Drawing Order contains the first batch of GDI+ 1.1 cacheable drawing primitives that comprise a rendering update sent by the server to the client. Support for GDI+ 1.1 rendering and primitive caching is specified in the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | Flags | | | | | | | | CacheType | | | | | | | | | | | | | | | |
| CacheIndex | | | | | | | | | | | | | | | | cbSize | | | | | | | | | | | | | | | |
| cbTotalSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| emfRecords (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as specified in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_GDIP\_CACHE\_FIRST (0x08).

**Flags (1 byte):** An 8-bit, unsigned integer. Flags indicating instructions on how to handle previous cache orders in relation to this one.

| Value | Meaning |
| --- | --- |
| GDIP\_REMOVE\_CACHEENTRY  0x01 | Remove the cache entry item at the index specified by **CacheIndex** before caching the one contained in this order. |

**CacheType (2 bytes):** A [GDI+ Cache Type (section 2.2.2.2.1.3.6.1.1)](#Section_9dee1f7b3989470dba973d78f4111099) structure. GDI+ cache in which to store the EMF+ Records contained in the **emfRecords** field.

**CacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the cache entry into which to write the primitive. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the GDI+ Graphics, Pen, Brush, Image and Image Attributes caches. The maximum number of entries allowed in each of these caches is specified by the **GdipCacheEntries** field of the Draw GDI+ Capability Set (section 2.2.1.3).

**cbSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the **emfRecords** field.

**cbTotalSize (4 bytes):** A 32-bit, unsigned integer. The cumulative size in bytes of the data in all of the **emfRecords** fields in this and subsequent [Draw GDI+ Cache Next](#Section_20da0b5e8cc94805998e03e9f216bade) and [Draw GDI+ Cache End](#Section_b44bd2b06c00454db182dbb841131c7a) Orders.

**emfRecords (variable):** A collection of EMF+ Records specified in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3. The size of the **emfRecords** field is given by the **cbSize** field.

Draw GDI+ Cache Next (DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER)

The Draw GDI+ Cache Next Alternate Secondary Drawing Order contains the second or subsequent batch of GDI+ 1.1 cacheable drawing primitives that comprise a rendering update sent by the server to the client. The first primitive in the sequence MUST have been transmitted with the [Draw GDI+ Cache First](#Section_53dbc9c54e3f4b17a256baf43c582146) Alternate Secondary Drawing Order. Support for GDI+ 1.1 rendering and primitive caching is specified in the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | Flags | | | | | | | | CacheType | | | | | | | | | | | | | | | |
| CacheIndex | | | | | | | | | | | | | | | | cbSize | | | | | | | | | | | | | | | |
| emfRecords (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_GDIP\_CACHE\_NEXT (0x09).

**Flags (1 byte):** An 8-bit, unsigned integer. Flags indicating instructions on how to handle previous cache orders in relation to this one.

| Value | Meaning |
| --- | --- |
| GDIP\_REMOVE\_CACHEENTRY  0x01 | Remove the cache entry item at the index specified by **CacheIndex** before caching the one contained in this order. |

**CacheType (2 bytes):** A [GDI+ Cache Type (section 2.2.2.2.1.3.6.1.1)](#Section_9dee1f7b3989470dba973d78f4111099) structure. GDI+ cache in which to store the EMF+ Records contained in the **emfRecords** field.

**CacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the cache entry into which to write the primitive. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the GDI+ Graphics, Pen, Brush, Image and Image Attributes caches. The maximum number of entries allowed in each of these caches is specified by the **GdipCacheEntries** field of the Draw GDI+ Capability Set (section 2.2.1.3).

**cbSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the **emfRecords** field.

**emfRecords (variable):** A collection of EMF+ Records specified in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3. The size of the **emfRecords** field is given by the **cbSize** field.

Draw GDI+ Cache End (DRAW\_GDIPLUS\_CACHE\_END\_ORDER)

The Draw GDI+ Cache End Alternate Secondary Drawing Order contains the final batch of GDI+ 1.1 cacheable drawing primitives that comprise a rendering update sent by the server to the client. Support for GDI+ 1.1 rendering and primitive caching is specified in the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | Flags | | | | | | | | CacheType | | | | | | | | | | | | | | | |
| CacheIndex | | | | | | | | | | | | | | | | cbSize | | | | | | | | | | | | | | | |
| cbTotalSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| emfRecords (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_GDIP\_CACHE\_END (0x0A).

**Flags (1 byte):** An 8-bit, unsigned integer. Flags indicating instructions on how to handle previous cache orders in relation to this one.

| Value | Meaning |
| --- | --- |
| GDIP\_REMOVE\_CACHEENTRY  0x01 | Remove the cache entry item at the index specified by **CacheIndex** before caching the one contained in this order. |

**CacheType (2 bytes):** A [GDI+ Cache Type (section 2.2.2.2.1.3.6.1.1)](#Section_9dee1f7b3989470dba973d78f4111099) structure. GDI+ cache in which to store the EMF+ Records contained in the **emfRecords** field.

**CacheIndex (2 bytes):** A 16-bit, unsigned integer. The index of the cache entry into which to write the primitive. This value MUST be greater than or equal to 0 and less than the maximum number of entries allowed in the GDI+ Graphics, Pen, Brush, Image and Image Attributes caches. The maximum number of entries allowed in each of these caches is specified by the **GdipCacheEntries** field of the Draw GDI+ Capability Set (section 2.2.1.3).

**cbSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the **emfRecords** field.

**cbTotalSize (4 bytes):** A 32-bit, unsigned integer. The cumulative size in bytes of the data in all of the **emfRecords** fields in this and previous [Draw GDI+ Cache Next](#Section_20da0b5e8cc94805998e03e9f216bade) and Draw GDI+ Cache End Orders.

**emfRecords (variable):** A collection of EMF+ Records as defined in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3. The size of the **emfRecords** field is given by the **cbSize** field.

Draw GDI+ First (DRAW\_GDIPLUS\_FIRST\_ORDER)

The Draw GDI+ First Alternate Secondary Drawing Order contains the first batch of GDI+ 1.1 drawing primitives that comprise a rendering update sent by the server to the client. Support for GDI+ 1.1 rendering is specified in the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | pad1Octet | | | | | | | | cbSize | | | | | | | | | | | | | | | |
| cbTotalSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbTotalEmfSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| emfRecords (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_GDIP\_FIRST (0x05).

**pad1Octet (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field MUST be ignored.

**cbSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the **emfRecords** field.

**cbTotalSize (4 bytes):** A 32-bit, unsigned integer. The cumulative size in bytes of the data in all of the **emfRecords** fields in this and subsequent Draw GDI+ Next and Draw GDI+ End orders.

**cbTotalEmfSize (4 bytes):** A 32-bit, unsigned integer. The cumulative size, in bytes, of the EMF+ record data (in this and subsequent [Draw GDI+ Next](#Section_b5deca1d5802428998cb87035e1ca454) and [Draw GDI+ End](#Section_657e77cd01d94eccaf45a69e0b48390f) Orders) that MUST be passed to the GDI+ subsystem for rendering. Because references to GDI+ primitives cached with Draw GDI+ Cache Orders (sections [2.2.2.2.1.3.6.2](#Section_53dbc9c54e3f4b17a256baf43c582146), [2.2.2.2.1.3.6.3](#Section_20da0b5e8cc94805998e03e9f216bade), and [2.2.2.2.1.3.6.4](#Section_b44bd2b06c00454db182dbb841131c7a)) can be contained in the EMF+ Records embedded in the **emfRecords** field, the total size of the EMF+ Records passed to the GDI+ subsystem for rendering can be larger than what was sent on the wire.

**emfRecords (variable):** A collection of EMF+ Records specified in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3. The size of the **emfRecords** field is given by the **cbSize** field. If the most significant bit of the **Size** field of an EMF+ Record is set, the EMF+ Record data contains a 16-bit cache index that MUST be used to retrieve a cached GDI+ 1.1 primitive from the appropriate GDI+ cache; the primitive MUST then be embedded in the EMF+ Record stream.

Draw GDI+ Next (DRAW\_GDIPLUS\_NEXT\_ORDER)

The Draw GDI+ Next Alternate Secondary Drawing Order contains the second or subsequent batch of GDI+ 1.1 drawing primitives that comprise a rendering update sent by the server to the client. The first primitive in the sequence MUST have been transmitted with the [Draw GDI+ First](#Section_03941c5e98d64f61ac74ddb307f1f317) Alternate Secondary Drawing Order. Support for GDI+ 1.1 rendering is specified in the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | pad1Octet | | | | | | | | cbSize | | | | | | | | | | | | | | | |
| emfRecords (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_GDIP\_NEXT (0x06).

**pad1Octet (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field MUST be ignored.

**cbSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the **emfRecords** field.

**emfRecords (variable):** Collection of EMF+ Records specified in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3. The size of the **emfRecords** field is given by the **cbSize** field. If the most significant bit of the **Size** field of an EMF+ Record is set, the EMF+ Record data contains a 16-bit cache index that MUST be used to retrieve a cached GDI+ 1.1 primitive from the appropriate GDI+ cache; the primitive MUST then be embedded in the EMF+ Record stream.

Draw GDI+ End (DRAW\_GDIPLUS\_END\_ORDER)

The Draw GDI+ End Alternate Secondary Drawing Order contains the final batch of GDI+ 1.1 drawing primitives that comprise a rendering update sent by the server to the client. Support for GDI+ 1.1 rendering is specified in the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | pad1Octet | | | | | | | | cbSize | | | | | | | | | | | | | | | |
| cbTotalSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbTotalEmfSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| emfRecords (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**header (1 byte):** An Alternate Secondary Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_GDIP\_END (0x07).

**pad1Octet (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field MUST be ignored.

**cbSize (2 bytes):** A 16-bit, unsigned integer. The size in bytes of the **emfRecords** field.

**cbTotalSize (4 bytes):** A 32-bit, unsigned integer. The cumulative size in bytes of the data in all of the **emfRecords** fields in this and previous [Draw GDI+ First](#Section_03941c5e98d64f61ac74ddb307f1f317) and [Draw GDI+ Next](#Section_b5deca1d5802428998cb87035e1ca454) Orders.

**cbTotalEmfSize (4 bytes):** A 32-bit, unsigned integer. The cumulative size, in bytes, of the EMF+ record data (in this and previous Draw GDI+ First and Draw GDI+ Next Orders) that MUST be passed to the GDI+ subsystem for rendering. Because references to GDI+ primitives cached with Draw GDI+ Cache Orders (sections [2.2.2.2.1.3.6.2](#Section_53dbc9c54e3f4b17a256baf43c582146), [2.2.2.2.1.3.6.3](#Section_20da0b5e8cc94805998e03e9f216bade), and [2.2.2.2.1.3.6.4](#Section_b44bd2b06c00454db182dbb841131c7a)) can be contained in the EMF+ Records embedded in the **emfRecords** field, the total size of the EMF+ Records passed to the GDI+ subsystem for rendering can be larger than what was sent on the wire.

**emfRecords (variable):** A collection of EMF+ Records specified in [[MS-EMFPLUS]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-EMFPLUS%5d.pdf) section 2.3. The size of the **emfRecords** field is given by the **cbSize** field. If the most significant bit of the **Size** field of an EMF+ Record is set, the EMF+ Record data contains a 16-bit cache index that MUST be used to retrieve a cached GDI+ 1.1 primitive from the appropriate GDI+ cache; the primitive MUST then be embedded in the EMF+ Record stream.

Frame Marker

The Frame Marker Alternate Secondary Drawing Order is used by the server to indicate to the client the beginning and end of a logical frame of graphics data. Breaking graphics data up into logical frames indicates to the client which orders SHOULD be rendered as a logical unit, hence helping to prevent [**screen tearing**](#gt_2aa588bc-ca7c-465b-aab7-26aec818b57b). Support for frame markers is specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| header | | | | | | | | action | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**header (1 byte):** An Alternate Secondary Drawing Order Header, as defined in section [2.2.2.2.1.3.1.1](#Section_b93614406e084a0697e76e1b7a4cc204). The embedded **orderType** field MUST be set to TS\_ALTSEC\_FRAME\_MARKER (0x0D).

**action (4 bytes):** A 32-bit, unsigned integer. Indicates the start or end of a logical frame.

| Value | Meaning |
| --- | --- |
| TS\_FRAME\_START  0x00000000 | Start of a logical frame of graphics data. All drawing orders from this point in the graphics stream are part of the same logical frame and can be rendered as one cohesive unit to prevent tearing. |
| TS\_FRAME\_END  0x00000001 | End of a logical frame of graphics data. |

#### Error Conditions

##### Client Bitmap Cache Error PDU

The Bitmap Cache Error PDU is sent by clients that support persistent bitmap caching and encounter caching errors. RDP 4.0, 5.0, 5.1, 5.2, 6.0, 6.1 and 7.0 servers honor up to five Bitmap Cache Error PDUs in a session.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| tpktHeader | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| x224Data | | | | | | | | | | | | | | | | | | | | | | | | mcsSDrq (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| securityHeader (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| bitmapCacheErrorPduData (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**tpktHeader (4 bytes):** A TPKT Header, as specified in [[T123]](http://go.microsoft.com/fwlink/?LinkId=90541) section 8.

**x224Data (3 bytes):** An X.224 Class 0 Data TPDU, as specified in [[X224]](http://go.microsoft.com/fwlink/?LinkId=90588) section 13.7.

**mcsSDrq (variable):** Variable-length, [**Packed Encoding Rules**](#gt_8fe1cf39-e1af-4f52-8616-0d9723b482dd) encoded (PER-encoded) MCS Domain PDU (DomainMCSPDU) that encapsulates an MCS Send Data Request structure (SDrq, choice 25 from DomainMCSPDU), as specified in [[T125]](http://go.microsoft.com/fwlink/?LinkId=90543) section 11.32 (the [**ASN.1**](#gt_2ae47996-2ca2-4ca7-9e80-8ba36f62051a) structure definitions are specified in [T125] section 7, parts 7 and 10). The **userData** field of the MCS Send Data Request PDU contains a Security Header and a [Bitmap Cache Error PDU Data (section 2.2.2.3.1.1)](#Section_484C7E89FC5F4F14B2D1F3D019D8A6B6) structure.

**securityHeader (variable):** Optional security header. The presence and format of the security header depends on the Encryption Level and Encryption Method selected by the server ([MS-RDPBCGR] sections 5.3.2 and 2.2.1.4.3). If the Encryption Level selected by the server is greater than ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is greater than ENCRYPTION\_METHOD\_NONE (0) then this field MUST contain one of the following headers:

* Non-FIPS Security Header ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.2.2) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_40BIT (0x00000001), ENCRYPTION\_METHOD\_56BIT (0x00000008), or ENCRYPTION\_METHOD\_128BIT (0x00000002).
* FIPS Security Header ([MS-RDPBCGR] section 2.2.8.1.1.2.3) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_FIPS (0x00000010).

If the Encryption Level selected by the server is ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is ENCRYPTION\_METHOD\_NONE (0), then this header MUST NOT be included in the PDU.

**bitmapCacheErrorPduData (variable):** The actual contents of the Bitmap Cache Error PDU, as specified in section 2.2.2.3.1.1.

###### Bitmap Cache Error PDU Data (TS\_BITMAP\_CACHE\_ERROR\_PDU)

The TS\_BITMAP\_CACHE\_ERROR\_PDU structure contains the contents of the Bitmap Cache Error PDU, which is essentially a Share Data Header (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.1.2) and an array of [Bitmap Cache Error Info (section 2.2.2.3.1.1.1)](#Section_2ecfb696fb2b4aca9202612828adade6) structures.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| shareDataHeader (18 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | NumInfoBlocks | | | | | | | | Pad1 | | | | | | | |
| Pad2 | | | | | | | | | | | | | | | | Info (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**shareDataHeader (18 bytes):** A Share Data Header (as specified in [MS-RDPBCGR] section 2.2.8.1.1.1.2) containing information on the packet. The type subfield of the **pduType** field of the Share Control Header within the Share Data Header MUST be set to PDUTYPE\_DATAPDU (7). The **pduType2** field of the Share Data Header MUST be set to PDUTYPE2\_BITMAPCACHE\_ERROR\_PDU (44).

**NumInfoBlocks (1 byte):** An 8-bit, unsigned integer. The number of Bitmap Cache Error Info (section 2.2.2.3.1.1.1) structures in the **Info** field.

**Pad1 (1 byte):** An 8-bit, unsigned integer. Padding. Values in this field are arbitrary and MUST be ignored.

**Pad2 (2 bytes):** A 16-bit, unsigned integer. Padding. Values in this field are arbitrary and MUST be ignored.

**Info (variable):** An array of Bitmap Cache Error Info (section 2.2.2.3.1.1.1) structures, each structure specifying what actions to take with each of the bitmap caches.

Bitmap Cache Error Info (TS\_BITMAP\_CACHE\_ERROR\_INFO)

The TS\_BITMAP\_CACHE\_ERROR\_INFO structure specifies what actions are to be taken on a particular bitmap cache when a caching error occurs.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| CacheID | | | | | | | | bBitField | | | | | | | | Pad | | | | | | | | | | | | | | | |
| NewNumEntries | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**CacheID (1 byte):** An 8-bit, unsigned integer. ID of the bitmap cache represented by this block.

**bBitField (1 byte):** An 8-bit, unsigned integer. A bit field containing several flags.

| Value | Meaning |
| --- | --- |
| BC\_ERR\_FLUSH\_CACHE  0x01 | Indicates that the contents of the cache MUST be emptied. |
| BC\_ERR\_NEWNUMENTRIES\_VALID  0x02 | Indicates that the **NewNumEntries** field is valid. If the BC\_ERR\_FLUSH\_CACHE (0x01) flag is not set, and the **NewNumEntries** field specifies a new non-zero size, the previous cache contents in the initial **NewNumEntries** cells MUST be preserved. |

**Pad (2 bytes):** A 16-bit, unsigned integer. Padding. Values in this field are ignored.

**NewNumEntries (4 bytes):** A 32-bit, unsigned integer. The new number of entries in the cache. This value MUST be less than or equal to the number of entries specified in the Revision 2 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.2).

##### Client Offscreen Bitmap Cache Error PDU

The Offscreen Bitmap Cache Error PDU is sent by clients that support offscreen bitmap caching and encounter caching errors.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| tpktHeader | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| x224Data | | | | | | | | | | | | | | | | | | | | | | | | mcsSDrq (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| securityHeader (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| offscreenBitmapCacheErrorPduData (22 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**tpktHeader (4 bytes):** A TPKT Header as specified in [[T123]](http://go.microsoft.com/fwlink/?LinkId=90541) section 8.

**x224Data (3 bytes):** An X.224 Class 0 Data TPDU, as specified in [[X224]](http://go.microsoft.com/fwlink/?LinkId=90588) section 13.7.

**mcsSDrq (variable):** Variable-length PER-encoded MCS Domain PDU (DomainMCSPDU) that encapsulates an MCS Send Data Request structure (SDrq, choice 25 from DomainMCSPDU), as specified in [[T125]](http://go.microsoft.com/fwlink/?LinkId=90543) section 11.32 (the ASN.1 structure definitions are specified in [T125] section 7, parts 7 and 10). The **userData** field of the MCS Send Data Request PDU contains a Security Header and an Offscreen Bitmap Cache Error PDU Data (section [2.2.2.3.2.1](#Section_F12C4D8B533548B6933E807480B237CE)) structure.

**securityHeader (variable):** Optional security header. The presence and format of the security header depends on the Encryption Level and Encryption Method selected by the server ([MS-RDPBCGR] sections 5.3.2 and 2.2.1.4.3). If the Encryption Level selected by the server is greater than ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is greater than ENCRYPTION\_METHOD\_NONE (0), then this field MUST contain one of the following headers:

* Non-FIPS Security Header ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.2.2) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_40BIT (0x00000001), ENCRYPTION\_METHOD\_56BIT (0x00000008), or ENCRYPTION\_METHOD\_128BIT (0x00000002).
* FIPS Security Header ([MS-RDPBCGR] section 2.2.8.1.1.2.3) if the Encryption Level selected by the server is ENCRYPTION\_METHOD\_FIPS (0x00000010).

If the Encryption Level selected by the server is ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is ENCRYPTION\_METHOD\_NONE (0), then this header MUST NOT be included in the PDU.

**offscreenBitmapCacheErrorPduData (22 bytes):** The actual contents of the Offscreen Bitmap Cache Error PDU, as specified in section 2.2.2.3.2.1.

###### Offscreen Bitmap Cache Error PDU Data (TS\_OFFSCRCACHE\_ERROR\_PDU)

The TS\_OFFSCRCACHE\_ERROR\_PDU structure contains the contents of the Offscreen Bitmap Cache Error PDU, which is essentially a Share Data Header (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.1.2) and a flags field.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| shareDataHeader (18 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | flags | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**shareDataHeader (18 bytes):** A Share Data Header (as specified in [MS-RDPBCGR] section 2.2.8.1.1.1.2) containing information on the packet. The type subfield of the **pduType** field of the Share Control Header within the Share Data Header MUST be set to PDUTYPE\_DATAPDU (7). The **pduType2** field of the Share Data Header MUST be set to PDUTYPE2\_OFFSCRCACHE\_ERROR\_PDU (46).

**flags (4 bytes):** A 32-bit, unsigned integer. Indicates the support for offscreen bitmap caching. This field MUST be set to **OC\_ERR\_FLUSH\_AND\_DISABLE\_OFFSCREEN** (0x00000001), which specifies that the offscreen cache MUST be flushed and that further offscreen bitmap caching MUST be disabled.

| Name | Value |
| --- | --- |
| OC\_ERR\_FLUSH\_AND\_DISABLE\_OFFSCREEN | 0x00000001 |

##### Client DrawNineGrid Cache Error PDU

The DrawNineGrid Cache Error PDU is sent by clients that support NineGrid bitmap caching and encounter caching errors.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| tpktHeader | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| x224Data | | | | | | | | | | | | | | | | | | | | | | | | mcsSDrq (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| securityHeader (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| drawNineGridErrorPduData (22 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**tpktHeader (4 bytes):** A TPKT Header, as specified in [[T123]](http://go.microsoft.com/fwlink/?LinkId=90541) section 8.

**x224Data (3 bytes):** An X.224 Class 0 Data TPDU, as specified in [[X224]](http://go.microsoft.com/fwlink/?LinkId=90588) section 13.7.

**mcsSDrq (variable):** Variable-length PER-encoded MCS Domain PDU (DomainMCSPDU) that encapsulates an MCS Send Data Request structure (SDrq, choice 25 from DomainMCSPDU), as specified in [[T125]](http://go.microsoft.com/fwlink/?LinkId=90543) section 11.32 (the ASN.1 structure definitions are specified in [T125] section 7, parts 7 and 10). The **userData** field of the MCS Send Data Request contains a Security Header and a DrawNineGrid Cache Error PDU Data (section [2.2.2.3.3.1](#Section_D88A4151E1C04BB4A77E897BCA29008B)) structure.

**securityHeader (variable):** Optional security header. The presence and format of the security header depends on the Encryption Level and Encryption Method selected by the server ([MS-RDPBCGR] sections 5.3.2 and 2.2.1.4.3). If the Encryption Level selected by the server is greater than ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is greater than ENCRYPTION\_METHOD\_NONE (0), then this field MUST contain one of the following headers:

* Non-FIPS Security Header ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.2.2) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_40BIT (0x00000001), ENCRYPTION\_METHOD\_56BIT (0x00000008), or ENCRYPTION\_METHOD\_128BIT (0x00000002).
* FIPS Security Header ([MS-RDPBCGR] section 2.2.8.1.1.2.3) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_FIPS (0x00000010).

If the Encryption Level selected by the server is ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is ENCRYPTION\_METHOD\_NONE (0), then this header MUST NOT be included in the PDU.

**drawNineGridErrorPduData (22 bytes):** The actual contents of the DrawNineGrid Cache Error PDU, as specified in section 2.2.2.3.3.1.

###### DrawNineGrid Cache Error PDU Data (TS\_DRAWNINEGRID\_ERROR\_PDU)

The TS\_DRAWNINEGRID\_ERROR\_PDU structure contains the contents of the DrawNineGrid Cache Error PDU, which is essentially a Share Data Header (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.1.2) and a flags field.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| shareDataHeader (18 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | flags | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**shareDataHeader (18 bytes):** A Share Data Header (as specified in [MS-RDPBCGR] section 2.2.8.1.1.1.2) containing information on the packet. The type subfield of the **pduType** field of the Share Control Header within the Share Data Header MUST be set to PDUTYPE\_DATAPDU (7). The **pduType2** field of the Share Data Header MUST be set to PDUTYPE2\_DRAWNINEGRID\_ERROR\_PDU (48).

**flags (4 bytes):** A 32-bit, unsigned integer. Indicates support for NineGrid bitmap caching. This field MUST be set to DNG\_ERR\_FLUSH\_AND\_DISABLE\_DRAWNINEGRID (0x00000001), which means that the NineGrid bitmap cache MUST be flushed and that further NineGrid bitmap caching MUST be disabled.

| Name | Value |
| --- | --- |
| DNG\_ERR\_FLUSH\_AND\_DISABLE\_DRAWNINEGRID | 0x00000001 |

##### Client GDI+ Error PDU

The GDI+ Error PDU is sent by clients that support GDI+ 1.1 rendering and encounter errors.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| tpktHeader | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| x224Data | | | | | | | | | | | | | | | | | | | | | | | | mcsSDrq (variable) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| securityHeader (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| gdiplusErrorPduData (22 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**tpktHeader (4 bytes):** A TPKT Header, as specified in [[T123]](http://go.microsoft.com/fwlink/?LinkId=90541) section 8.

**x224Data (3 bytes):** An X.224 Class 0 Data TPDU, as specified in [[X224]](http://go.microsoft.com/fwlink/?LinkId=90588) section 13.7.

**mcsSDrq (variable):** Variable-length PER-encoded MCS Domain PDU (DomainMCSPDU) that encapsulates an MCS Send Data Request structure (SDrq, choice 25 from DomainMCSPDU), as specified in [[T125]](http://go.microsoft.com/fwlink/?LinkId=90543) section 11.32 (the ASN.1 structure definitions are specified in [T125] section 7, parts 7 and 10). The **userData** field of the MCS Send Data Request contains a Security Header and a GDI+ Error PDU Data (section [2.2.2.3.4.1](#Section_43C4679794654D21AEB2A1DCD79578B3)) structure.

**securityHeader (variable):** Optional security header. The presence and format of the security header depends on the Encryption Level and Encryption Method selected by the server ([MS-RDPBCGR] sections 5.3.2 and 2.2.1.4.3). If the Encryption Level selected by the server is greater than ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is greater than ENCRYPTION\_METHOD\_NONE (0) then this field MUST contain one of the following headers:

* Non-FIPS Security Header ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.2.2) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_40BIT (0x00000001), ENCRYPTION\_METHOD\_56BIT (0x00000008), or ENCRYPTION\_METHOD\_128BIT (0x00000002).
* FIPS Security Header ([MS-RDPBCGR] section 2.2.8.1.1.2.3) if the Encryption Method selected by the server is ENCRYPTION\_METHOD\_FIPS (0x00000010).

If the Encryption Level selected by the server is ENCRYPTION\_LEVEL\_NONE (0) and the Encryption Method selected by the server is ENCRYPTION\_METHOD\_NONE (0), then this header MUST NOT be included in the PDU.

**gdiplusErrorPduData (22 bytes):** The actual contents of the GDI+ Error PDU, as specified in section 2.2.2.3.4.1.

###### GDI+ Error PDU Data (TS\_DRAWGDIPLUS\_ERROR\_PDU)

The TS\_DRAWGDIPLUS\_ERROR\_PDU structure contains the contents of the GDI+ Error PDU, which is essentially a Share Data Header (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.8.1.1.1.2) and a flags field.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| shareDataHeader (18 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | flags | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**shareDataHeader (18 bytes):** A Share Data Header containing information on the packet (see [MS-RDPBCGR] section 2.2.8.1.1.1.2). The type subfield of the **pduType** field of the Share Control Header within the Share Data Header MUST be set to PDUTYPE\_DATAPDU (7). The **pduType2** field of the Share Data Header MUST be set to PDUTYPE2\_DRAWGDIPLUS\_ERROR\_PDU (49).

**flags (4 bytes):** A 32-bit, unsigned integer. Indicates support for GDI+ 1.1 rendering. This field MUST be set to GDIPLUS\_ERR\_FLUSH\_AND\_DISABLE\_DRAWGDIPLUS (0x00000001), which means that the GDI+ drawing MUST be flushed and further GDI+ 1.1 rendering primitives MUST be disabled (implying that future drawings will be sent as bitmaps).

#### Bulk Data Compression

##### RDP 6.1 Compressed Data (RDP61\_COMPRESSED\_DATA)

The RDP61\_COMPRESSED\_DATA structure stores chain-compressed data that has been processed by the RDP 6.1 Level-1 Compression Engine and possibly the RDP 5.0 Level-2 Compression Engine (see section [3.1.8.2.1](#Section_1a454e38738d4d9caf41d4db0e5f2425) for a description of chained compression and [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8.4.2 for a description of RDP 5.0 bulk compression).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Level1ComprFlags | | | | | | | | Level2ComprFlags | | | | | | | | MatchCount (optional) | | | | | | | | | | | | | | | |
| MatchDetails (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Literals (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Level1ComprFlags (1 byte):** An 8-bit, unsigned integer. Level-1 compressor flags.

| Value | Meaning |
| --- | --- |
| L1\_PACKET\_AT\_FRONT  0x04 | The level-1 history buffer MUST be reinitialized (by filling it with zeros). |
| L1\_NO\_COMPRESSION  0x02 | No compression was performed. In this case, the MatchCount and MatchDetails fields MUST NOT be present. The Literals field MUST be present. |
| L1\_COMPRESSED  0x01 | Compression with the level-1 compressor was performed and the MatchCount and MatchDetails fields MUST be present and contain at least one match. The Literals field MUST also be present. |
| L1\_INNER\_COMPRESSION  0x10 | Indicates that additional level-2 compression has been performed on the level-1 compressor output and that the Level2ComprFlags field contains valid data and MUST be processed. |

See sections [3.1.8.2.2.2](#Section_0683eadbd52a4c5ca03946123d983de6) and [3.1.8.2.3](#Section_c33884dec58c43b883261f93076a0504) for more details regarding these flags.

**Level2ComprFlags (1 byte):** An 8-bit, unsigned integer. Level-2 compressor flags.

| Value | Meaning |
| --- | --- |
| PACKET\_COMPR\_TYPE\_64K  0x01 | Indicates that RDP 5.0 bulk compression (see [MS-RDPBCGR] section 3.1.8.4.2) was used. |
| PACKET\_COMPRESSED  0x20 | The data in the MatchCount, MatchDetails, and Literals fields has been compressed with the level-2 compressor. |
| PACKET\_AT\_FRONT  0x40 | The decompressed data MUST be placed at the beginning of the level-2 history buffer. |
| PACKET\_FLUSHED  0x80 | The level-2 history buffer MUST be reinitialized (by filling it with zeros). |

See [MS-RDPBCGR] sections 3.1.8.2.1 and 3.1.8.3 for more details regarding these flags.

The **Level2ComprFlags** field MUST be ignored if the L1\_INNER\_COMPRESSION flag (0x10) is not set.

**MatchCount (2 bytes):** A 16-bit, unsigned integer. Contains the number of RDP 6.1 Match Details structures (see section [2.2.2.4.1.1](#Section_01f927654cd844bf95f212b91d127dd5)) present in the MatchDetails field. This field MUST be present if the L1\_COMPRESSED flag (0x01) is set in the Level1ComprFlags field.

**MatchDetails (variable):** A variable-length field. Contains an array of RDP 6.1 Match Details structures (see section 2.2.2.4.1.1). This field MUST NOT be present if the MatchCount field is not present. If the MatchCount field is present and contains a value greater than zero, the MatchDetails field MUST be present, and the number of RDP 6.1 Match Details structures that it contains is given by the MatchCount field.

**Literals (variable):** A variable-length field. Contains raw noncompressed literals. The size of the Literals field is given by the equation, CompressedDataSize - (2 + MatchCountSize + MatchDetailsSize).

###### RDP 6.1 Match Details (RDP61\_MATCH\_DETAILS)

The RDP61\_MATCH\_DETAILS structure encapsulates all of the details, which describes a compression match in a history buffer. (See section [3.1.8.2.2](#Section_18c798368dfa4860bf96e70e402a5f24) for a description of how matches are employed within the RDP 6.1 Compression Engine.)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| MatchLength | | | | | | | | | | | | | | | | MatchOutputOffset | | | | | | | | | | | | | | | |
| MatchHistoryOffset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**MatchLength (2 bytes):** A 16-bit, unsigned integer. The length in bytes of the match in the level-1 history buffer.

**MatchOutputOffset (2 bytes):** A 16-bit, unsigned integer. The relative offset into the output buffer where the match MUST be copied to from the history buffer.

**MatchHistoryOffset (4 bytes):** A 32-bit, unsigned integer. The byte offset into the current level-1 history buffer specifying where the match begins.

#### Bitmap Compression

##### RDP 6.0 Bitmap Compressed Bitmap Stream (RDP6\_BITMAP\_STREAM)

The RDP6\_BITMAP\_STREAM structure contains a stream of bitmap data compressed using RDP 6.0 Bitmap Compression techniques (section [3.1.9](#Section_0d2e1a03e12346b2b2b8ed730a794ae4)). Depending on the compression techniques employed, the bitmap data is represented using the [**AYCoCg**](#gt_74e654be-4f30-46ff-982a-e7c497e3152b) or [**ARGB**](#gt_a99d8c7d-4340-403d-b965-1f1e0b7defb1) [**color space**](#gt_5d3fb5ea-c686-4d3b-b0ff-aef33fe1aee5) (section [3.1.9.1.2](#Section_ef530a0a03d8482f989b57a1036797b2)).

Compressed bitmap data is sent encapsulated in a Bitmap Update ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.1.3.1.2), Fast-Path Bitmap Update ([MS-RDPBCGR] section 2.2.9.1.2.1.2 ), Cache Bitmap - Revision 1 (section [2.2.2.2.1.2.2)](#Section_c9365b320be547c9af1afeba2ea1ee9f) Secondary Drawing Orders, or Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3)](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40) Secondary Drawing Orders. In all of these cases, the data is encapsulated inside a Bitmap Data structure ([MS-RDPBCGR] section 2.2.9.1.1.3.1.2.2).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| FormatHeader | | | | | | | | AlphaPlane (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LumaOrRedPlane (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| OrangeChromaOrGreenPlane (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GreenChromaOrBluePlane (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pad (optional) | | | | | | | |

**FormatHeader (1 byte):** An 8-bit, unsigned integer. This field contains a 1-byte, bit-packed update header as follows.

The format of the update header byte is described by the following bitmask diagram.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| CLL | | | A | B | C | Reserved | |

**CLL (3 bits):** A 3-bit, unsigned integer field that indicates the Color Loss Level (section [3.1.9.1.4](#Section_b4462e6422864e779c5e2046fc490cc0)). If **CLL** is set to 0, the color space used is ARGB. Otherwise, **CLL** MUST be in the range 1 to 7 (inclusive), and the color space used is AYCoCg.

**A - CS (1 bit):** A 1-bit field that indicates whether chroma subsampling is being used (section [3.1.9.1.3](#Section_57507b9441e641ffa35ec00211945ceb)). If **CS** is equal to 1, chroma subsampling is being used, and the **CLL** field MUST be greater than 0, as chroma subsampling applies only to the AYCoCg color space.

**B - RLE (1 bit):** A 1-bit field. If **RLE** is equal to 1, RDP 6.0 RLE is used to compress the [**color planes**](#gt_78d62be3-d07a-447e-ab51-fa8be5560fe0) (section [3.1.9.2](#Section_c5f101a07e674f3ca678d9760ce8921c)). If not, **RLE** is equal to 0, and the color plane is sent uncompressed.

**C - NA (1 bit):** A 1-bit field. Indicates if an alpha plane is present. If **NA** is equal to 1, there is no alpha plane. The values of the alpha plane are then assumed to be 0xFF (fully opaque), and the bitmap data contains only three color planes. If **NA** is equal to 0, the alpha plane is sent as the first color plane.

**Reserved (2 bits):** A 2-bit, unsigned integer field. Reserved for future use.

**AlphaPlane (variable):** A variable-length field that contains the alpha plane. If the **RLE** subfield in the FormatHeader indicates that all of the color planes are RLE compressed (section 3.1.9.2), this field contains an RDP 6.0 RLE Segments (section [2.2.2.5.1.1)](#Section_623172526dff4b5682909e8b5513665b) structure. Otherwise, it contains the raw bytes of the color plane.

**LumaOrRedPlane (variable):** A variable-length field that contains the luma plane (AYCoCg color space) or the red plane (ARGB color space). If the **CLL** subfield of the FormatHeader is greater than 0, the AYCoCg color space MUST be used. Otherwise, the ARGB color space MUST be used.

If the **RLE** subfield in the FormatHeader indicates that all of the color planes are RLE compressed (section 3.1.9.2), this field contains an RDP 6.0 RLE Segments (section 2.2.2.5.1.1) structure. Otherwise, it contains the raw bytes of the color plane.

**OrangeChromaOrGreenPlane (variable):** A variable-length field that contains the orange chroma plane (AYCoCg color space) or the green plane (ARGB color space). If the **CLL** subfield of the FormatHeader is greater than 0, the AYCoCg color space MUST be used. Otherwise, the ARGB color space MUST be used.

If the **RLE** subfield in the FormatHeader indicates that all of the color planes are RLE compressed (section 3.1.9.2), this field contains an RDP 6.0 RLE Segments (section 2.2.2.5.1.1) structure. Otherwise, it contains the raw bytes of the color plane.

Depending on the values of the **CLL** and **CS** subfields of the FormatHeader (in the case of the AYCoCg color space), the orange chroma plane may have been transformed by color loss reduction (section 3.1.9.1.4) and chroma subsampling (section 3.1.9.1.3).

**GreenChromaOrBluePlane (variable):** A variable-length field that contains the green chroma plane (AYCoCg color space) or the blue plane (ARGB color space). If the **CLL** subfield of the FormatHeader is greater than 0, the AYCoCg color space MUST be used. Otherwise, the ARGB color space MUST be used.

If the **RLE** subfield in the FormatHeader indicates that all of the color planes are RLE compressed (section 3.1.9.2), this field contains an RDP 6.0 RLE Segments (section 2.2.2.5.1.1) structure. Otherwise, it contains the raw bytes of the color plane.

Depending on the values of the **CLL** and **CS** subfields of the FormatHeader (in the case of the AYCoCg color space), the green chroma plane may have been transformed by color loss reduction (section 3.1.9.1.4) and chroma subsampling (section 3.1.9.1.3).

**Pad (1 byte):** An 8-bit, unsigned integer containing padding values that MUST be ignored. This optional field is only present if the **RLE** subfield of the **FormatHeader** field is zero.

###### RDP 6.0 RLE Segments (RDP6\_RLE\_SEGMENTS)

The RDP6\_RLE\_SEGMENTS structure contains the run-length encoded contents of a color plane and consists of a collection of [RDP6\_RLE\_SEGMENT](#Section_f7e4c717669e4f318b34e8f5ab2e107e) structures.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| segments (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**segments (variable):** This field contains a variable-length array of RDP 6.0 RLE Segment (section 2.2.2.5.1.2) structures.

###### RDP 6.0 RLE Segment (RDP6\_RLE\_SEGMENT)

The RDP6\_RLE\_SEGMENT structure encodes an RLE segment that contains a RAW and RUN component (section [3.1.9.2](#Section_c5f101a07e674f3ca678d9760ce8921c)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| controlByte | | | | | | | | rawValues (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**controlByte (1 byte):** An 8-bit, unsigned integer. Contains the RAW and RUN components of an RDP 6.0 RLE segment (section 3.1.9.2). The **controlByte** field MUST contain a nonzero value.

The format of the **controlByte** field is described by the following bitmask diagram.

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

**nRunLength (4 bits):** A 4-bit, unsigned integer field. The number of times the last RAW byte in the **rawValues** field is repeated (known as the run-length).

Because a RUN MUST be a sequence of at least three values (section 3.1.9.2), the values 1 and 2 are used in the **nRunLength** field to encode extra long RUN sequences of more than 16 values:

* If the **nRunLength** field is set to 1, the actual run-length is 16 plus the value in **cRawBytes**. On decode, the number of RAW bytes in the **rawValues** field is assumed to be zero. This gives a maximum run-length of 31 values.
* If the **nRunLength** field is set to 2, the actual run-length is 32 plus the value in **cRawBytes**. On decode, the number of RAW bytes in the **rawValues** field is assumed to be zero. This gives a maximum run-length of 47 values.

**cRawBytes (4 bits):** A 4-bit, unsigned integer field. The number of RAW bytes in the **rawValues** field.

**rawValues (variable):** A variable-length field that contains the actual RAW bytes representing a portion of the color plane being encoded. If the bytes belong to the first scan line, they represent absolute values. Otherwise, the bytes represent delta values (section 3.1.9.2).

# Protocol Details

## Common Details

### Abstract Data Model

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

**Note**  It is possible to implement the following conceptual data by using a variety of techniques as long as the implementation produces external behavior that is consistent with that described in this document.

#### Caches

All of the RDP caches are managed by the server. The server determines the items to cache and in what cache entries these items MUST be stored. These items, and the metadata describing the cache and cache entry, are sent to the client using an appropriate drawing order. The layout of most of the caches is usually specified by using a suitable capability set.

##### Bitmap Caches

Bitmap caches are used by the client and server to store graphic bitmaps. Each bitmap cache holds bitmaps of a specified size in pixels (known as the "tile size"). If a bitmap does not fit into a single cache entry, the server uses a tiling algorithm to divide the bitmap into tiles that will fit into the cache entries so that they can be stored separately into the cache.

The Cache Bitmap - Revision 1 (section [2.2.2.2.1.2.2](#Section_c9365b320be547c9af1afeba2ea1ee9f)), Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40)), and Cache Bitmap - Revision 3 (section [2.2.2.2.1.2.8](#Section_710cf5c26a0c4cea8d62fa4f639d7c5a)) Secondary Drawing Orders are used to update the contents of the client-side bitmap cache.

There are two versions of bitmap caches. Revision 1 bitmap caches are used in association with the Cache Bitmap – Revision 1 Secondary Drawing Order and only support memory-based caching. Revision 2 bitmap caches are used in association with the both the Cache Bitmap – Revision 2 and Cache Bitmap – Revision 3 Secondary Drawing Orders and support persistent disk caching (in addition to memory caching) by associating a 64-bit key (derived from a cryptographic hash of the bitmap contents) with each bitmap. These 64-bit keys SHOULD be sent to the server on subsequent connections (using the Persistent Key List PDU specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.1.17.1) to initialize the persistent disk cache. The server maintains its own cache of bitmap keys that it has already sent to the client (section [3.3.1.1](#Section_4f9f384944f947f0aab6112d28ca6197)).

When using the Revision 2 bitmap caches, the client is able to request that the server delays forcing it to cache a bitmap to disk until the bitmap has been used more than once. This is implemented by requesting that the server maintain a Bitmap Cache Wait List (see section [3.3.1.3](#Section_fb7b28f140c94535bd60b4fd345d94c8)). Once a particular bitmap has been encountered by the server more than once, it MUST instruct the client to cache it (see section 2.2.2.2.1.2.3).

Support for the Revision 2 bitmap caches is advertised by the server by using the Bitmap Cache Host Support Capability Set ([MS-RDPBCGR] section 2.2.7.2.1). Client support for the Revision 1 bitmap caches is advertised by using the Revision 1 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.1) while support for the Revision 2 bitmap caches is advertised by using the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2).

The Revision 1 bitmap cache defaults to 1500 KB of memory cache for 8 bpp, 3000 KB for 16 bpp, 4500 KB for 24 bpp, and 6000 KB for 32 bpp. The Revision 2 bitmap cache defaults to 10 MB of persistent cache storage for 8 bpp, 20 MB for 16 bpp, 30 MB for 24 bpp, and 40 MB for 32 bpp. The actual layout of the bitmap caches is specified in the Bitmap Cache Capability Sets ([MS-RDPBCGR] section 2.2.7.1.4).

The following table shows the default Revision 1 bitmap cache layout.

| Bitmap cache ID | Tile size (in pixels) | Cache entry size | Number of cache entries |
| --- | --- | --- | --- |
| 0 | 16 x 16 | 256 bytes for 8 bpp  512 bytes for 16 bpp  768 bytes for 24 bpp  1024 bytes for 32 bpp | 120 |
| 1 | 32 x 32 | 1024 bytes for 8 bpp  2048 bytes for 16 bpp  3072 bytes for 24 bpp  4096 bytes for 32 bpp | 120 |
| 2 | 64 x 64 | 4096 bytes for 8 bpp  8192 bytes for 16 bpp  12288 bytes for 24 bpp  16384 bytes for 32 bpp | 337 |

The following table shows the default Revision 2 bitmap cache layout with persistent storage enabled for cache 2.

| Bitmap cache ID | Tile size (in pixels) | Cache entry size | Number of cache entries |
| --- | --- | --- | --- |
| 0 | 16 x 16 | 256 bytes for 8 bpp  512 bytes for 16 bpp  768 bytes for 24 bpp  1024 bytes for 32 bpp | 120 |
| 1 | 32 x 32 | 1024 bytes for 8 bpp  2048 bytes for 16 bpp  3072 bytes for 24 bpp  4096 bytes for 32 bpp | 120 |
| 2 | 64 x 64 | 4096 bytes for 8 bpp  8192 bytes for 16 bpp  12288 bytes for 24 bpp  16384 bytes for 32 bpp | 2547 for 8 bpp  2553 for 16 bpp  2555 for 24 bpp  2556 for 32 bpp |

If persistent bitmap caching is enabled, the client SHOULD enumerate the entries in its local persistent bitmap cache to obtain the 64-bit bitmap keys for all of the stored bitmaps, and then send one or more Persistent Key List PDUs that contains this data to the server ([MS-RDPBCGR] section 2.2.1.17).

Bitmap images stored in the bitmap caches are rendered using the MemBlt (section [2.2.2.2.1.1.2.9](#Section_84c2ec2ff776405b9b486894a28b1b14)) and Mem3Blt (section [2.2.2.2.1.1.2.10](#Section_3aa21a4a031c4a3f8fc5159068f30237)) Primary Drawing Orders.

##### Glyph and Fragment Caches

Glyph caching supports 10 glyph caches and 1 fragment cache to store bitmaps of font characters in memory. Glyphs are first cached in the glyph caches before being displayed. A fragment is a set of glyphs defined in terms of glyph indices.

The Cache Glyph - Revision 1 (section [2.2.2.2.1.2.5)](#Section_6dc06088b1124f4fb0bcb97bba621a0c) and Cache Glyph - Revision 2 (section [2.2.2.2.1.2.6)](#Section_ffc80435136e442e908540ca1d27a413) Secondary Drawing Orders are used to update the glyph caches. The [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) and [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7) Primary Drawing Orders consume glyphs from the glyph caches and also populate the Fragment Cache. The [FastGlyph (section 2.2.2.2.1.1.2.15)](#Section_839e963ce24a42adb4bdeb108340629c) Primary Drawing Order is used to update the glyph caches with a single glyph and to encode the same glyph at a specified position.

The actual layout of the glyph and fragment caches is specified in the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

The following table shows the default Glyph Cache layout.

| Glyph cache ID | Cache entry size (in bytes) | Number of cache entries |
| --- | --- | --- |
| 0 | 4 | 254 |
| 1 | 4 | 254 |
| 2 | 8 | 254 |
| 3 | 8 | 254 |
| 4 | 16 | 254 |
| 5 | 32 | 254 |
| 6 | 64 | 254 |
| 7 | 128 | 254 |
| 8 | 256 | 254 |
| 9 | 256 | 64 |

There is only one fragment cache, and by default it has 256 entries with 256 bytes as the entry size.

##### Color Table Cache

The Color Table Cache is used to reduce bandwidth by caching color palettes. The existence of the Color Table Cache is implicitly tied to support for the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders. If support for these orders is advertised in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the existence of a color table cache with entries for six palettes is implied when palettized color is being used. Each of the six cached color tables holds 256 color mappings, initialized by the [Cache Color Table (section 2.2.2.2.1.2.4)](#Section_49fad30181714f38a68a7394baeb7b15) Secondary Drawing Order. Cached color tables are used exclusively by the MemBlt (section 2.2.2.2.1.1.2.9) and Mem3Blt (section 2.2.2.2.1.1.2.10) Primary Drawing Orders.

##### Brush Caches

There are two brush caches: a mono brush cache and a color brush cache. Each cache can hold 64 brush entries. The size of a mono brush is 16 bytes, and the size of a color brush is 64 bytes for 8 bpp and 192 bytes for 24 bpp. Support for brush caching is specified by using the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7). The [Cache Brush (section 2.2.2.2.1.2.7)](#Section_9e7027c4d9894cb09674e2e36c95ebdd) Secondary Drawing Order is used to populate the brush caches.

##### Offscreen Bitmap Cache

The Offscreen Bitmap Cache is used to store writable offscreen bitmap surfaces. There is only one cache for all offscreen bitmaps. The bitmap sizes are variable and depend on the dimensions specified by the [Create Offscreen Bitmap (section 2.2.2.2.1.3.2)](#Section_427758ff4bac4833b388d634cc512d0c) Alternate Secondary Drawing Order.

The total cache size and cache entries are capped by the values specified in the Offscreen Bitmap Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9). The default size of the cache is 2.5 MB for 8 bpp, 5 MB for 16 bpp, or 7.5 MB for 24 bpp. The default number of allowed cache entries is 100.

The Create Offscreen Bitmap (section 2.2.2.2.1.3.2) and [Switch Surface (section 2.2.2.2.1.3.3)](#Section_ada4d8a9421e48c0a0ed7384fa3cf061) Alternate Secondary Drawing Orders are used to manipulate the offscreen cache. Images stored in the Offscreen Bitmap Cache are rendered with the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders.

##### NineGrid Bitmap Cache

The NineGrid bitmap cache is used to store NineGrid-compliant bitmaps. There is only one cache for all NineGrid bitmaps. The individual bitmap sizes are variable and depend on the bitmap dimensions.

The total cache size and cache entries are capped by the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780). The default size of the cache is 2.5 MB for 8 bpp, 5 MB for 16 bpp, or 7.5 MB for 24 bpp. The default number of allowed cache entries is 256.

The [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38), [Stream Bitmap First (section 2.2.2.2.1.3.5.1)](#Section_b714a7c97fd54621a397349a4ef3e06c), and [Stream Bitmap Next (section 2.2.2.2.1.3.5.2)](#Section_dfe099ffa894406393a7fbe64e7cb85e) Alternate Secondary Drawing Orders are used to populate the NineGrid Bitmap Cache. Individual bitmaps within the cache are rendered with the [DrawNineGrid (section 2.2.2.2.1.1.2.21)](#Section_44bd6a68a5b74a38980706ba4ab4a72a) Primary Drawing Order. This primary drawing order applies a NineGrid algorithm to the bitmap before rendering it (for an example rendering, see section [4.4](#Section_06ba35beca674fd1937e7a0ac678b63a)).

##### GDI+ Caches

The GDI+ caches are used to store Graphics, Brush, Pen, Image, and Image Attributes primitives. The total cache sizes and cache entries permitted are specified by using the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4).

### Timers

None.

### Initialization

None.

### Higher-Layer Triggered Events

None.

### Processing Events and Sequencing Rules

There are no common message processing events or sequencing rules. Specification of message processing events and rules is deferred to sections [3.2.5](#Section_4cbdeb652bc7428fa9afdecc916dee21) and [3.3.5](#Section_2686d734c19b4bccb65ea433afcd72e6).

### Timer Events

None.

### Other Local Events

None.

### Bulk Data Compression

#### RDP 6.0

RDP version 6.0 supports an extension to the compression techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8. This extension is called "RDP 6.0 Bulk Compression" (RDP6.0-BC) and is only supported for server-to-client traffic.

##### Abstract Data Model

The shared state necessary to support the transmission and reception of RDP6.0-BC compressed data between a client and server requires a history buffer and a current offset into the history buffer (HistoryOffset). The size of the history buffer is 65,536 bytes. Any data that is being compressed MUST be smaller in size than the history buffer. The HistoryOffset MUST start initialized to zero, while the history buffer MUST be filled with zeros. After it has been initialized, the entire history buffer is immediately regarded as valid.

In addition to the history buffer and HistoryOffset, a small cache MUST also be managed by the client and server endpoints. This cache is referred to as the OffsetCache and is used to store the last four unique copy-offsets encountered during data compression (copy-offsets are described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8.1). This saves on bandwidth in cases where there are many repeated copy-offsets. Whenever the history buffer is initialized or reinitialized, the OffsetCache MUST be emptied.

When compressing data, the sender MUST first check that the uncompressed data can be inserted into the history buffer at the position in the history buffer given by the HistoryOffset. If the data will not fit into the history buffer (the sum of the HistoryOffset and the size of the uncompressed data exceeds the size of the history buffer), then the history buffer MUST be slid back by half its size and the HistoryOffset MUST be reset to the middle of the history buffer (offset 32768). If the data will fit into the history buffer, the sender endpoint inserts the uncompressed data at the position in the history buffer given by the HistoryOffset, and then advances the HistoryOffset by the amount of data added.

As the receiver endpoint decompresses the data, it inserts the decompressed data at the position in the history buffer given by its local copy HistoryOffset. If a slide-back occurs, the sender endpoint MUST notify the target receiver so it can reset its local state. In this way, the sender and receiver endpoints maintain an exact replica of the history buffer and HistoryOffset.

##### Compressing Data

Compression using RDP6.0-BC is based on the principles outlined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8.2 with literals and copy-tuples being encoded using the scheme described in section [3.1.8.1.4.3](#Section_fb37ff5718264811892b42266fdd5301). Also, due to the slide-back behavior outlined in section [3.1.8.1.1](#Section_08c389bf1a3c40e8af0f6e32654fd8f8), the meaning of the PACKET\_AT\_FRONT flag (0x40) has changed. This flag MUST still be set in conjunction with the PACKET\_COMPRESSED (0x20) flag; however, it is now used to indicate that the history buffer MUST be slid back by half its size and that the HistoryOffset MUST be reset to the middle of the history buffer.

##### Decompressing Data

Decompression using RDP6.0-BC is based on the principles specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8.3 with the compressed stream being decoded using the scheme described in section [3.1.8.1.4.4](#Section_f652cccf9be24f14a025d3247efb7f5f). Also, due to the slide-back behavior described in section [3.1.8.1.1](#Section_08c389bf1a3c40e8af0f6e32654fd8f8), the interpretation of the PACKET\_AT\_FRONT flag (0x40) has changed. If this flag is set, the decompressor MUST slide the history buffer back by half its size and reset the HistoryOffset to the middle of the history buffer.

##### Wire Format

The compressed stream consists of a bit-encoded sequence of literals, <copy-offset, length-of-match> tuples (also known as copy-tuples), and a terminating End-of-Stream (EOS) marker.

Literals, copy-offsets, and EOS marker values are Huffman encoded to produce 293 individual LiteralOrEosOrCopyOffset variable-length bit-codes (possibly followed by extra bits of information). Each LiteralOrEosOrCopyOffset code is optionally followed by 1 of 32 possible Huffman encoded length-of-match codes (LengthOfMatch) that is present, depending on the contents of the previous LiteralOrEosOrCopyOffset code (the LengthOfMatch code is also possibly followed by extra bits of information).

The convention used on the stream is that the sender MUST output multi-byte fields in little-endian byte order with the most significant bit of each byte in the first position. For example:

* 0xAABBCC is output on the wire as 0xCC, followed by 0xBB, followed by 0xAA.
* 0x8001 is output on the wire as 0x01, followed by 0x80.
* 0x01 is output on the wire as 0x01.

For more information on Huffman encoding, see [SAYOOD] sections 4.1 to 4.7.

###### Literal, EOS, and Copy-Offset Tables

The length, in bits, for each of the 293 Huffman-encoded LiteralOrEosOrCopyOffset codes are listed in the following table (HuffLengthLEC).

Codes with an index value of 0 to 255 represent a single literal character with that value. Code index 256 indicates that the end of the encoding has been reached. Index values 257 to 288 must be translated by subtracting 257 to find the index value 0 to 31 in the CopyOffsetBitsLUT and CopyOffsetBaseLUT tables, to find the number of bits and base values for a copy operation. Index values 289 to 292 must be translated by subtracting 289 to find the index value 0 to 3 to identify one of the four most recently-used offsets.

| Code index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0x6 | 0x6 | 0x6 | 0x7 | 0x7 | 0x7 | 0x7 | 0x7 |
| 8 | 0x7 | 0x7 | 0x7 | 0x8 | 0x8 | 0x8 | 0x8 | 0x8 |
| 16 | 0x8 | 0x8 | 0x9 | 0x8 | 0x9 | 0x9 | 0x9 | 0x9 |
| 24 | 0x8 | 0x8 | 0x9 | 0x9 | 0x9 | 0x9 | 0x9 | 0x9 |
| 32 | 0x8 | 0x9 | 0x9 | 0xa | 0x9 | 0x9 | 0x9 | 0x9 |
| 40 | 0x9 | 0x9 | 0x9 | 0xa | 0x9 | 0xa | 0xa | 0xa |
| 48 | 0x9 | 0x9 | 0xa | 0x9 | 0xa | 0x9 | 0xa | 0x9 |
| 56 | 0x9 | 0x9 | 0xa | 0xa | 0x9 | 0xa | 0x9 | 0x9 |
| 64 | 0x8 | 0x9 | 0x9 | 0x9 | 0x9 | 0xa | 0xa | 0xa |
| 72 | 0x9 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 80 | 0x9 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 88 | 0xa | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 96 | 0x8 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 104 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 112 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 120 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0x9 |
| 128 | 0x7 | 0x9 | 0x9 | 0xa | 0x9 | 0xa | 0xa | 0xa |
| 136 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 144 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 152 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 160 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 168 | 0xa | 0xa | 0xa | 0xd | 0xa | 0xa | 0xa | 0xa |
| 176 | 0xa | 0xa | 0xb | 0xa | 0xa | 0xa | 0xa | 0xa |
| 184 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 192 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0x9 | 0xa |
| 200 | 0xa | 0xa | 0xa | 0xa | 0x9 | 0xa | 0xa | 0xa |
| 208 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 216 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 224 | 0x9 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa |
| 232 | 0xa | 0xa | 0xa | 0xa | 0xa | 0xa | 0x9 | 0xa |
| 240 | 0x8 | 0x9 | 0x9 | 0xa | 0x9 | 0xa | 0xa | 0xa |
| 248 | 0x9 | 0xa | 0xa | 0xa | 0x9 | 0x9 | 0x8 | 0x7 |
| 256 | 0xd | 0xd | 0x7 | 0x7 | 0xa | 0x7 | 0x7 | 0x6 |
| 264 | 0x6 | 0x6 | 0x6 | 0x5 | 0x6 | 0x6 | 0x6 | 0x5 |
| 272 | 0x6 | 0x5 | 0x6 | 0x6 | 0x6 | 0x6 | 0x6 | 0x6 |
| 280 | 0x6 | 0x6 | 0x6 | 0x6 | 0x6 | 0x6 | 0x6 | 0x6 |
| 288 | 0x8 | 0x5 | 0x6 | 0x7 | 0x7 |  |  |  |

Table 1: Bit lengths for the 293 Huffman-encoded LiteralOrEosOrCopyOffset codes

For example, it can be determined from the previous table that the 0th Huffman-encoded LiteralOrEosOrCopyOffset code has a length of 6 bits, and the 131st Huffman-encoded LiteralOrEosOrCopyOffset code has a length of 10 (0x0A) bits.

Using the canonical Huffman algorithm ([SAYOOD] section 4.2.4), the Huffman codebook table shown in the following table (HuffCodeLEC) can be obtained. The bit lengths in the previous table MUST be used to isolate the appropriate bits.

| Code index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0x0004 | 0x0024 | 0x0014 | 0x0011 | 0x0051 | 0x0031 | 0x0071 | 0x0009 |
| 8 | 0x0049 | 0x0029 | 0x0069 | 0x0015 | 0x0095 | 0x0055 | 0x00d5 | 0x0035 |
| 16 | 0x00b5 | 0x0075 | 0x001d | 0x00f5 | 0x011d | 0x009d | 0x019d | 0x005d |
| 24 | 0x000d | 0x008d | 0x015d | 0x00dd | 0x01dd | 0x003d | 0x013d | 0x00bd |
| 32 | 0x004d | 0x01bd | 0x007d | 0x006b | 0x017d | 0x00fd | 0x01fd | 0x0003 |
| 40 | 0x0103 | 0x0083 | 0x0183 | 0x026b | 0x0043 | 0x016b | 0x036b | 0x00eb |
| 48 | 0x0143 | 0x00c3 | 0x02eb | 0x01c3 | 0x01eb | 0x0023 | 0x03eb | 0x0123 |
| 56 | 0x00a3 | 0x01a3 | 0x001b | 0x021b | 0x0063 | 0x011b | 0x0163 | 0x00e3 |
| 64 | 0x00cd | 0x01e3 | 0x0013 | 0x0113 | 0x0093 | 0x031b | 0x009b | 0x029b |
| 72 | 0x0193 | 0x0053 | 0x019b | 0x039b | 0x005b | 0x025b | 0x015b | 0x035b |
| 80 | 0x0153 | 0x00d3 | 0x00db | 0x02db | 0x01db | 0x03db | 0x003b | 0x023b |
| 88 | 0x013b | 0x01d3 | 0x033b | 0x00bb | 0x02bb | 0x01bb | 0x03bb | 0x007b |
| 96 | 0x002d | 0x027b | 0x017b | 0x037b | 0x00fb | 0x02fb | 0x01fb | 0x03fb |
| 104 | 0x0007 | 0x0207 | 0x0107 | 0x0307 | 0x0087 | 0x0287 | 0x0187 | 0x0387 |
| 112 | 0x0033 | 0x0047 | 0x0247 | 0x0147 | 0x0347 | 0x00c7 | 0x02c7 | 0x01c7 |
| 120 | 0x0133 | 0x03c7 | 0x0027 | 0x0227 | 0x0127 | 0x0327 | 0x00a7 | 0x00b3 |
| 128 | 0x0019 | 0x01b3 | 0x0073 | 0x02a7 | 0x0173 | 0x01a7 | 0x03a7 | 0x0067 |
| 136 | 0x00f3 | 0x0267 | 0x0167 | 0x0367 | 0x00e7 | 0x02e7 | 0x01e7 | 0x03e7 |
| 144 | 0x01f3 | 0x0017 | 0x0217 | 0x0117 | 0x0317 | 0x0097 | 0x0297 | 0x0197 |
| 152 | 0x0397 | 0x0057 | 0x0257 | 0x0157 | 0x0357 | 0x00d7 | 0x02d7 | 0x01d7 |
| 160 | 0x03d7 | 0x0037 | 0x0237 | 0x0137 | 0x0337 | 0x00b7 | 0x02b7 | 0x01b7 |
| 168 | 0x03b7 | 0x0077 | 0x0277 | 0x07ff | 0x0177 | 0x0377 | 0x00f7 | 0x02f7 |
| 176 | 0x01f7 | 0x03f7 | 0x03ff | 0x000f | 0x020f | 0x010f | 0x030f | 0x008f |
| 184 | 0x028f | 0x018f | 0x038f | 0x004f | 0x024f | 0x014f | 0x034f | 0x00cf |
| 192 | 0x000b | 0x02cf | 0x01cf | 0x03cf | 0x002f | 0x022f | 0x010b | 0x012f |
| 200 | 0x032f | 0x00af | 0x02af | 0x01af | 0x008b | 0x03af | 0x006f | 0x026f |
| 208 | 0x018b | 0x016f | 0x036f | 0x00ef | 0x02ef | 0x01ef | 0x03ef | 0x001f |
| 216 | 0x021f | 0x011f | 0x031f | 0x009f | 0x029f | 0x019f | 0x039f | 0x005f |
| 224 | 0x004b | 0x025f | 0x015f | 0x035f | 0x00df | 0x02df | 0x01df | 0x03df |
| 232 | 0x003f | 0x023f | 0x013f | 0x033f | 0x00bf | 0x02bf | 0x014b | 0x01bf |
| 240 | 0x00ad | 0x00cb | 0x01cb | 0x03bf | 0x002b | 0x007f | 0x027f | 0x017f |
| 248 | 0x012b | 0x037f | 0x00ff | 0x02ff | 0x00ab | 0x01ab | 0x006d | 0x0059 |
| 256 | 0x17ff | 0x0fff | 0x0039 | 0x0079 | 0x01ff | 0x0005 | 0x0045 | 0x0034 |
| 264 | 0x000c | 0x002c | 0x001c | 0x0000 | 0x003c | 0x0002 | 0x0022 | 0x0010 |
| 272 | 0x0012 | 0x0008 | 0x0032 | 0x000a | 0x002a | 0x001a | 0x003a | 0x0006 |
| 280 | 0x0026 | 0x0016 | 0x0036 | 0x000e | 0x002e | 0x001e | 0x003e | 0x0001 |
| 288 | 0x00ed | 0x0018 | 0x0021 | 0x0025 | 0x0065 |  |  |  |

Table 2: Huffman codebook for the 293 Huffman-encoded LiteralOrEosOrCopyOffset codes

For example, it can be determined from the HuffCodeLEC table that the 0th Huffman-encoded LiteralOrEosOrCopyOffset code has a value of 0x0004. Applying the bit-length information from the HuffLengthLEC table, it can be determined that the Huffman code MUST be 6 bits in length. Hence the final code in binary MUST be 000100.

As another example, it can be determined from the HuffCodeLEC table that the 131st Huffman-encoded LiteralOrEosOrCopyOffset code has a value of 0x02a7. Applying the bit-length information from the HuffLengthLEC table, it can be determined that the Huffman code MUST be 10 bits in length. Hence the final code in binary MUST be 1010100111.

The two lookup tables in the following table (CopyOffsetBitsLUT and CopyOffsetBaseLUT) are used during encoding and decoding of the copy-offset for a given copy-tuple to determine the extra bits that will follow the LiteralOrEosOrCopyOffset code.

| Index | CopyOffsetBitsLUT | CopyOffsetBaseLUT |
| --- | --- | --- |
| 0 | 0 | 1 |
| 1 | 0 | 2 |
| 2 | 0 | 3 |
| 3 | 0 | 4 |
| 4 | 1 | 5 |
| 5 | 1 | 7 |
| 6 | 2 | 9 |
| 7 | 2 | 13 |
| 8 | 3 | 17 |
| 9 | 3 | 25 |
| 10 | 4 | 33 |
| 11 | 4 | 49 |
| 12 | 5 | 65 |
| 13 | 5 | 97 |
| 14 | 6 | 129 |
| 15 | 6 | 193 |
| 16 | 7 | 257 |
| 17 | 7 | 385 |
| 18 | 8 | 513 |
| 19 | 8 | 769 |
| 20 | 9 | 1025 |
| 21 | 9 | 1537 |
| 22 | 10 | 2049 |
| 23 | 10 | 3073 |
| 24 | 11 | 4097 |
| 25 | 11 | 6145 |
| 26 | 12 | 8193 |
| 27 | 12 | 12289 |
| 28 | 13 | 16385 |
| 29 | 13 | 24577 |
| 30 | 14 | 32769 |
| 31 | 14 | 49153 |

Table 3: Bit count and base value lookup tables to encode and decode copy-offset values

###### Length-of-Match Tables

The length, in bits, for each of the 0 to 31 Huffman-encoded LengthOfMatch codes are listed in the following table (HuffLengthL).

| LengthOfMatch Code | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0x4 | 0x2 | 0x3 | 0x4 | 0x3 | 0x4 | 0x4 | 0x5 |
| 8 | 0x4 | 0x5 | 0x5 | 0x6 | 0x6 | 0x7 | 0x7 | 0x8 |
| 16 | 0x7 | 0x8 | 0x8 | 0x9 | 0x9 | 0x8 | 0x9 | 0x9 |
| 24 | 0x9 | 0x9 | 0x9 | 0x9 | 0x9 | 0x9 | 0x9 | 0x9 |

Table 4: Bit lengths for the 32 Huffman-encoded LengthOfMatch codes

For example, it can be determined from the previous table that the 0th Huffman-encoded LengthOfMatch code has a length of 4 bits while the 21st Huffman-encoded LengthOfMatch code has a length of 8 bits.

Using the canonical Huffman algorithm ([SAYOOD] section 4.2.4), the Huffman codebook table shown in the following table (HuffCodeL) can be obtained. The bit lengths in the previous table MUST be used to mask out the appropriate bits.

| Code index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0x0001 | 0x0000 | 0x0002 | 0x0009 | 0x0006 | 0x0005 | 0x000d | 0x000b |
| 8 | 0x0003 | 0x001b | 0x0007 | 0x0017 | 0x0037 | 0x000f | 0x004f | 0x006f |
| 16 | 0x002f | 0x00ef | 0x001f | 0x005f | 0x015f | 0x009f | 0x00df | 0x01df |
| 24 | 0x003f | 0x013f | 0x00bf | 0x01bf | 0x007f | 0x017f | 0x00ff | 0x01ff |

Table 5: Huffman codebook for the 32 Huffman-encoded LengthOfMatch codes

For example, it can be determined from the HuffCodeL table that the 0th Huffman-encoded LengthOfMatch code has a value of 0x0001. Applying the bit-length information from the HuffLengthL table, it can be determined that the Huffman code MUST be 4 bits in length. Hence the final code in binary MUST be 0001.

As another example, it can be determined from the HuffCodeL table that the 21st Huffman-encoded LengthOfMatch code has a value of 0x009f. Applying the bit-length information from the HuffLengthL table, it can be determined that the Huffman code MUST be 8 bits in length. Hence the final code in binary MUST be 10011111.

The two lookup tables in the following table (LoMBitsLUT and LoMBaseLUT) are used during encoding and decoding of the length-of-match for a given copy-tuple to determine the extra bits that will follow the LengthOfMatch code. Note that the minimum match length that can be encoded is 2, as opposed to RDP 4.0 and 5.0 bulk data compression where the minimum is 3 (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) sections 3.1.8.2 and 3.1.8.4.1.2.2).

| Index | LoMBitsLUT | LoMBaseLUT |
| --- | --- | --- |
| 0 | 0 | 2 |
| 1 | 0 | 3 |
| 2 | 0 | 4 |
| 3 | 0 | 5 |
| 4 | 0 | 6 |
| 5 | 0 | 7 |
| 6 | 0 | 8 |
| 7 | 0 | 9 |
| 8 | 1 | 10 |
| 9 | 1 | 12 |
| 10 | 1 | 14 |
| 11 | 1 | 16 |
| 12 | 2 | 18 |
| 13 | 2 | 22 |
| 14 | 2 | 26 |
| 15 | 2 | 30 |
| 16 | 3 | 34 |
| 17 | 3 | 42 |
| 18 | 3 | 50 |
| 19 | 3 | 58 |
| 20 | 4 | 66 |
| 21 | 4 | 82 |
| 22 | 4 | 98 |
| 23 | 4 | 114 |
| 24 | 6 | 130 |
| 25 | 6 | 194 |
| 26 | 8 | 258 |
| 27 | 8 | 514 |
| 28 | 14 | 2 |
| 29 | 14 | 2 |

Table 6: Bit count and base value lookup tables to encode and decode length-of-match values

###### Encoding the Logically Compressed Stream

The following flow chart describes how literals and copy-tuples in a logically compressed stream are encoded.

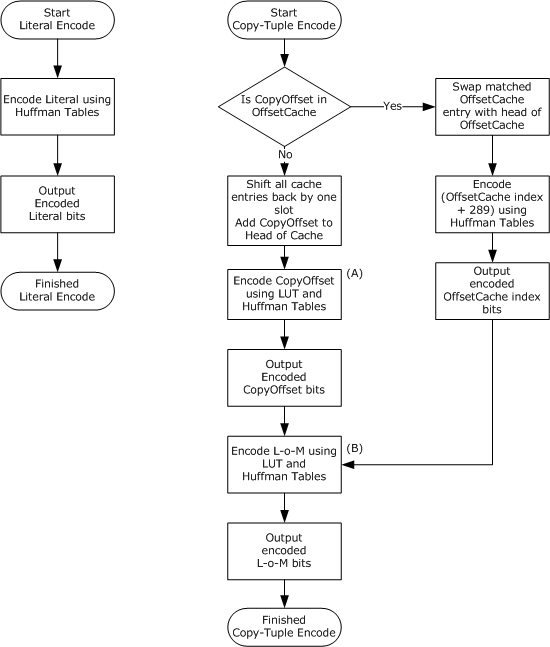


Figure : Encoding a logically compressed stream

Literals are merely encoded using the Huffman Tables HuffCodeLEC (table 2) and HuffLengthLEC (table 1). Copy-tuples are encoded using the four lookup tables in table 3 and table 6 (CopyOffsetBaseLUT, CopyOffsetBitsLUT, LoMBaseLUT, and LoMBitsLUT) and the Huffman Tables (for the algorithmic details, see sections [3.1.8.1.4.3.1](#Section_fc9e9fd31a104c26994f8a29088020e4) and [3.1.8.1.4.3.2](#Section_29a7fdaf1e1549f6adec1184c4358c47)).

Encoding the Copy-Offset

Encoding of the copy-offset is shown in the previous figure by the Action A item. The following describes the algorithm for encoding the copy-offset.

1. LUTIndex = IndexOfEqualOrSmallerEntry(CopyOffset + 1, CopyOffsetBaseLUT)
2. HuffmanIndex = LUTIndex + 257
3. OutputBits(HuffCodeLEC[HuffmanIndex], HuffLengthLEC[HuffmanIndex])
4. ExtraBitsLength = CopyOffsetBitsLUT[LUTIndex]
5. ExtraBits = CopyOffset & ((2 ^ ExtraBitsLen) - 1)
6. OutputBits(ExtraBits, ExtraBitsLength)

The IndexOfEqualOrSmallerEntry function searches through the specified LUT table and returns the index of the entry that contains a value of equal or lesser value than the first parameter. The OutputBits function outputs the bits specified by the first parameter in the appropriate order (the number of bits to output is given by the second parameter). "^" is the exponentiation operator, and "&" is the bitwise AND operator.

Examples of Copy-Offset Encoding

1. Encoding a copy-offset of 8.
2. LUTIndex = 6
3. HuffmanIndex = 6 + 257 = 263
4. OutputBits(0x0034, 6) = bin:110100
5. ExtraBitsLength = 2
6. ExtraBits = 8 & ((2 ^ 2) - 1) = 8 & 3 = bin:1000 & bin:0011 = 0
7. OutputBits(0x00, 2) = bin:00
8. Encoding a copy-offset of 11.
9. LUTIndex = 6
10. HuffmanIndex = 6 + 257 = 263
11. OutputBits(0x0034, 6) = bin:110100
12. ExtraBitsLength = 2
13. ExtraBits = 11 & ((2 ^ 2) - 1) = 11 & 3 = bin:1011 & bin:0011 = 3
14. OutputBits(0x03, 2) = bin:11

Encoding the Length-of-Match

Encoding of the length-of-match is shown in the previous figure by the Action B item. The following describes the algorithm for encoding the length-of-match.

1. LUTIndex = IndexOfEqualOrSmallerEntry(LoM, LoMBaseLUT)
2. OutputBits(HuffCodeL[LUTIndex], HuffLengthL[LUTIndex])
3. ExtraBitsLength = LoMBitsLUT[LUTIndex]
4. ExtraBits = (LoM - 2) & ((2 ^ ExtraBitsLength) - 1)
5. OutputBits (ExtraBits, ExtraBitsLength)

The definitions of the functions used in this pseudocode are the same as those described in section [3.1.8.1.4.3.1](#Section_fc9e9fd31a104c26994f8a29088020e4).

Examples of Length-of-Match Encoding

1. Encoding a length-of-match of 2 bytes.
2. LUTIndex = 0
3. HuffmanCode = 0x0001
4. OutputBits(0x0001, 4) = bin:0001
5. ExtraBitsLength = 0
6. No extra bits to output
7. Encoding a length-of-match of 117 bytes.
8. LUTIndex = 23
9. HuffmanCode = 0x01df
10. OutputBits(0x01df, 9) = bin:111011111
11. ExtraBitsLength = 4
12. ExtraBits = (117 - 2) & ((2 ^ 4) - 1) = 115 & 15 = bin:1110011 & bin:1111 = 3
13. OutputBits(0x03, 4) = bin:0011

###### Decoding a Compressed Stream

The following flow chart describes how the data in a compressed stream is decoded.

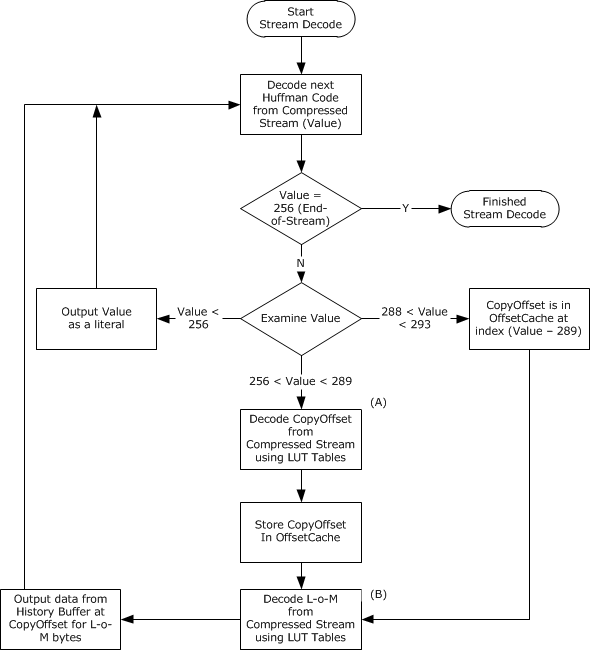


Figure : Decoding a compressed stream

Decoded literals are merely placed on to the output stream. However, decoded values representing copy-offset and length-of-match items require further processing by using the four lookup tables in table 3 and table 6 (CopyOffsetBaseLUT, CopyOffsetBitsLUT, LoMBaseLUT, and LoMBitsLUT) (for the algorithmic details, see sections [3.1.8.1.4.4.1](#Section_db23341a2d384530b526d93f6bce6368) and [3.1.8.1.4.4.2](#Section_e47f20650f984d69b90220fba439b5fc)).

Decoding the Copy-Offset

Decoding of the copy-offset is shown in the previous figure by the Action A item. The following describes the algorithm for decoding a copy-offset.

1. LUTIndex = DecodedHuffmanCode - 257
2. BaseLUT = CopyOffsetBaseLUT[LUTIndex]
3. BitsLUT = CopyOffsetBitsLUT[LUTIndex]
4. StreamBits = ReadBitsFromCompressedStream(BitsLUT)
5. CopyOffset = BaseLUT + StreamBits - 1

The ReadBitsFromCompressedStream function reads the number of bits specified by the parameter from the compressed stream.

Examples of Copy-Offset Decoding

1. Decoding a Huffman code of 0x0034 followed by 2 bits of value 0.
2. DecodedHuffmanCode = 263
3. LUTIndex = 263 - 257 = 6
4. BaseLUT = 9
5. BitsLUT = 2
6. StreamBits = ReadBitsFromCompressedStream(2) = 0x00
7. CopyOffset = 9 + 0 - 1 = 8
8. Decoding a Huffman code of 0x0034 followed by 2 bits of value 3.
9. DecodedHuffmanCode = 263
10. LUTIndex = 263 - 257 = 6
11. BaseLUT = 9
12. BitsLUT = 2
13. StreamBits = ReadBitsFromCompressedStream(2) = 0x03
14. CopyOffset = 9 + 3 - 1 = 11

Decoding the Length-of-Match

Decoding of the length-of-match is shown in the previous figure by the Action B item. The following describes the algorithm for decoding a length-of-match.

1. HuffmanCode = ReadNextHuffmanCodeFromCompressedStream()
2. LUTIndex = DecodeHuffmanCode(HuffmanCode, HuffmanCodeTable)
3. BaseLUT = LoMBaseLUT[LUTIndex]
4. BitsLUT = LoMBitsLUT[LUTIndex]
5. StreamBits = ReadBitsFromCompressedStream(BitsLUT)
6. LoM = BaseLUT + StreamBits

The ReadNextHuffmanCodeFromCompressedStream function reads the next Huffman code from the compressed stream, and the DecodeHuffmanCode function decodes the Huffman code given by the first parameter using the Huffman codebook table specified by the second parameter. The definitions of any remaining functions used in this pseudocode are the same as those described in section [3.1.8.1.4.4.1](#Section_db23341a2d384530b526d93f6bce6368).

Examples of Length-of-Match Decoding

1. Decoding a Huffman code of 0x0001 followed by 0 bits.
2. HuffmanCode = 0x0001
3. LUTIndex = 0
4. BaseLUT = 2
5. BitsLUT = 0
6. StreamBits = ReadBitsFromCompressedStream(0) = 0x00
7. LoM = 2 + 0 = 2
8. Decoding a Huffman code of 0x01df followed by 4 bits of value 3.
9. HuffmanCode = 0x01df
10. LUTIndex = 23
11. BaseLUT = 114
12. BitsLUT = 4
13. StreamBits = ReadBitsFromCompressedStream(4) = 0x03
14. LoM = 114 + 3 = 117

#### RDP 6.1

RDP 6.1 supports an extension to the compression techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8. This extension is called "RDP 6.1 Bulk Compression" (RDP6.1-BC) and is supported only for server-to-client traffic. The RDP6.1-BC compressor is unique in that it is a chainable compressor, where the output of the compressor can be passed as input into another compressor to further compress the data stream. This is possible because RDP6.1-BC is a byte-level stream compressor and does not perform bit-level compression. The RDP6.1-BC compressor is referred to as the "level-1" compressor and the chained RDP 5.0 bulk compressor (described in [MS-RDPBCGR] section 3.1.8) is known as the "level-2" compressor.

##### Abstract Data Model

The shared state necessary to support the transmission and reception of RDP6.1-BC compressed data between a client and server requires a level-1 history buffer (HistoryBuffer) and a current offset into the history buffer (HistoryOffset). The size of the history buffer is fixed at 2,000,000 bytes. Any single block of data that is being compressed by a compliant compressor MUST be smaller in size than 16,383 bytes. The HistoryOffset MUST start initialized to zero, while the history buffer MUST be filled with zeros. After it has been initialized, the entire history buffer is immediately regarded as valid.

The shared state necessary to support the level-2 RDP 5.0 bulk compressor is described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8.

The level-1 compressor attempts to describe large matches (lengths of up to 16,382 bytes or offsets of up to 2,000,000 bytes) using RDP61\_MATCH\_DETAILS (section [2.2.2.4.1.1](#Section_01f927654cd844bf95f212b91d127dd5)) structures. The matches could have been found anywhere in the 2,000,000-byte history buffer, including data realized earlier within the same block. Any such matches are encoded in the **MatchDetails** field of the RDP61\_COMPRESSED\_DATA (section [2.2.2.4.1](#Section_4b2427d8f80048229a3d82cf1ccd3ce7)) structure, and all remaining data is collapsed into the **Literals** field. The match count, details, and literals data are then presented to the level-2 compressor for more granular compression.

When compressing data, the sender MUST first check that the uncompressed data can be inserted into the level-1 history buffer at the position in the history buffer given by the HistoryOffset. If the data will not fit into the history buffer (the sum of the HistoryOffset and the size of the uncompressed data exceeds the size of the history buffer), then the HistoryOffset MUST be reset to the start of the history buffer (offset 0). If the data will fit into the history buffer, then the sender endpoint inserts the uncompressed data at the position in the history buffer given by the HistoryOffset, and then advances the HistoryOffset by the amount of data added.

If data compression results in an expansion of the data size (that is, the size of the compressed data plus any additional headers is larger than the original data size), then the original uncompressed data MUST instead be passed directly to the level-2 compressor (this is known as "compressor chaining"), and the L1\_NO\_COMPRESSION flag (see section [3.1.8.2.2.2](#Section_0683eadbd52a4c5ca03946123d983de6)) MUST be set in the level-1 compression header (see section 2.2.2.4.1).

Level-2 compression could result in an expansion of the data size. In this case, the PACKET\_COMPRESSED flag (see [MS-RDPBCGR] sections 3.1.8.2.1 and 3.1.8.3) MUST NOT be set in the appropriate level-2 compression header, and the level-2 data is passed unencoded. To ensure that the history buffers in both compressors remain synchronized, the compression flags of both compressors (if both compressors are used and the L1\_INNER\_COMPRESSION flag (0x10) is set) MUST be sent to the receiving endpoint. If the level-2 compressor state is out of sync with the decompressor (for example, if compression is attempted but expansion occurs), then the level-2 compressor MUST reset its internal state, and the level-2 compression flags MUST include the PACKET\_FLUSHED flag when the next packet with PACKET\_COMPRESSED is sent, in order to cause the decompressor to reset its internal state.

The receiving endpoint checks the L1\_INNER\_COMPRESSION flag (0x10), and if this flag is set decompresses the input packet according to the rules outlined in [MS-RDPBCGR] section 3.1.8 for RDP 5.0 bulk compression. The output from this decompression phase then becomes the raw data for the level-1 RDP6.1-BC compressor. As a result, the receiver decompresses the data again using the level-1 decompressor and inserts the final decompressed data at the position in the level-1 history buffer given by its local copy of HistoryOffset. If a reset of the level-1 or level-2 history offset occurs, then the sender endpoint MUST notify the target receiver so that it can also reset its local state. In this way, the sender and receiver endpoints maintain an exact replica of the level-1 (and level-2) history buffer(s) and HistoryOffset(s).

The flowchart in the following figure illustrates the process of compressing data by chaining two compressors.

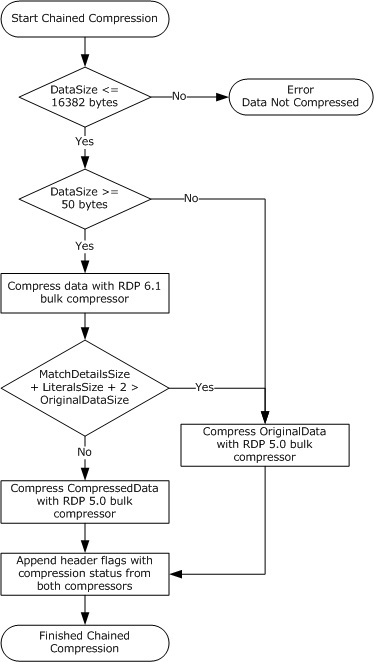


Figure : Chained compression with the RDP 6.1 and RDP 5.0 bulk compressors

The flowchart in following figure describes the how the decompressors are chained during decompression.

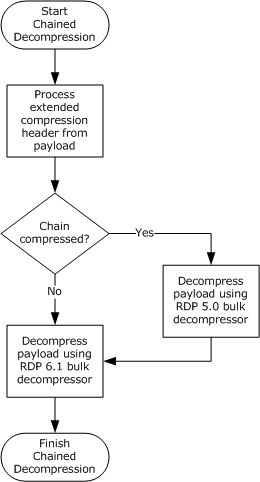


Figure : Chained decompression with the RDP 6.1 and RDP 5.0 bulk decompressors

##### Compressing Data

Compression using RDP6.1-BC is based on the principles outlined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8 with literals and copy-tuples (matches) being encoded using the scheme described in section [2.2.2.4.1](#Section_4b2427d8f80048229a3d82cf1ccd3ce7). The RDP6.1-BC compressor is stream-based and, as such, is able to include copy-tuples (referred to as "history matches") that have just been added to the current packet being processed; that is, as data is appended to the local history buffer, it can immediately be referenced in the next match. Matches MUST be in stream-order so that a match does not refer to any data not yet received by the target endpoint. Note that every encoded match incurs an overhead of 8 bytes when it is described by the RDP 6.1 Match Details (section [2.2.2.4.1.1](#Section_01f927654cd844bf95f212b91d127dd5)) structure. For this reason, an RDP 6.1 compliant compressor SHOULD exclude matches that result in expansion.

The following flowchart describes how the RDP6.1-BC compression algorithm operates.

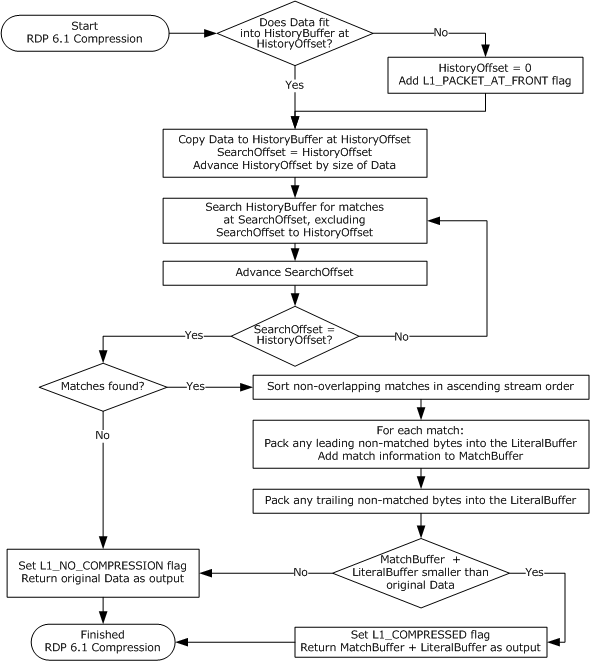


Figure : The RDP 6.1 bulk compression algorithm

###### Data Compression Example

The example in this section is merely illustrative of the RDP 6.1 compression techniques presented in section [3.1.8.2.2](#Section_18C798368DFA4860BF96E70E402A5F24). The minimum recommended match length is ignored and data expansion does occur when adding the RDP 6.1 Compressed Data (section [2.2.2.4.1](#Section_4B2427D8F80048229A3D82CF1CCD3CE7)) structure.

Consider the following stream data.

1. a b c d e f g h i j

With an empty history buffer, this data would translate as being all literals. The history buffer would appear as follows.

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j

The resultant output data would be as follows.

1. a b c d e f g h i j

Now, assume that the following data is added to the stream to be compressed.

1. k l m n o d e f g h i j k l a b c d u

The compressor would again append this to its history buffer (or add to the start of the history buffer if not enough space is available). The history buffer would then appear as follows.

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j k l m n o d e f g h i j k l a b c d u

The output from the compressor would become as follows.

1. k l m n o
2. [MatchHistoryOffset@3, MatchLength=9, MatchOutputOffset@5]
3. [MatchHistoryOffset@0, MatchLength=4, MatchOutputOffset@14]
4. u

The first match includes both literals from the first packet and literals in the beginning of the current packet. The important condition is that any match can contain only literals that have been sent previously in the stream and still exist at the time of compression in its history buffer. Note that the RDP6.1-BC copy-tuple encoding scheme not only specifies where the match begins in the history buffer (MatchHistoryOffset) and its length in bytes (MatchLength) but also its relative position in the decompressed output stream. This extra information enables an on-the-wire format that allows the matches and literals to be grouped independently rather than in line with each other in the compressed stream.

The following is a simplified version of the RDP6.1-BC wire format (described in section 2.2.2.4.1) for the compressor output previously described.

1. [2 matches]
2. [MatchHistoryOffset@3, MatchLength=9, MatchOutputOffset@5]
3. [MatchHistoryOffset@0, MatchLength=4, MatchOutputOffset@14]
4. k l m n o u

Notice that the two matches are grouped together and are not present in line with the literals in the stream.

###### Setting Compression and Extended Compression Flags

Once data has been compressed, the RDP6.1-BC compressor (the level-1 compressor) MUST call (or chain) another compressor (the level-2 compressor) to further compress the compressed output. The L1\_INNER\_COMPRESSION flag (0x10) MUST be set in the Level1ComprFlags field (see section [2.2.2.4.1](#Section_4b2427d8f80048229a3d82cf1ccd3ce7)) if a level-2 compressor is called.

If the level-1 compressor determines that applying compression will result in an expansion of the data size, it MAY instead pass the original data that it was asked to compress directly to the level-2 compressor. If the RDP6.1-BC compressor compresses the original data it MUST set the L1 COMPRESSED flag (0x01). If the RDP6.1-BC compressor reverts to the original data (no compression is possible) it MUST set the L1\_NO\_COMPRESSION flag (0x02). Either one of these flags MUST be set if the RDP6.1-BC compressor updates the level-1 history buffer. This ensures that the receiver is able to update its level-1 history buffer accordingly.

If either the L1\_COMPRESSED (0x01) or L1\_NO\_COMPRESSION (0x02) flags are set (that is, the RDP6.1-BC compressor has touched its level-1 history buffer), then the PDU which encapsulates the level-1 output data MUST set the PACKET\_COMPRESSED flag (0x20) in the header field appropriate to the type of data payload, such as Fast-Path output data (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.2.1), virtual channel data (see [MS-RDPBCGR] section 2.2.6.1), or Slow-Path data (see [MS-RDPBCGR] section 2.2.9.1.1). Hence, it is valid for the PACKET\_COMPRESSED flag (0x20) to be set when the L1\_NO\_COMPRESSION flag (0x02) is also set.

If there is not enough space in the level-1 history buffer to add the data to be compressed at the current history offset, then the history buffer offset MUST be reset to the start position (0) before the data is inserted. If a reset of this type occurs, then the level-1 L1\_PACKET\_AT\_FRONT flag (0x04) MUST be set to notify the receiver that an update to the level-1 history buffer offset is required. Unlike RDP6.0-BC, this is a simple reset of the history offset and no sliding window semantics are involved (see section [3.1.8.1.2](#Section_506a6581d43e440c98b29a48bd9a4974) for a description of RDP6.0-BC semantics).

The level-1 compression flags are produced by performing a logical OR operation of one or more of the following flags.

| Compression flag | Meaning |
| --- | --- |
| L1\_PACKET\_AT\_FRONT  0x04 | The level-1 history buffer MUST be reinitialized (by filling it with zeros). After it has been reinitialized, the entire history buffer is immediately regarded as valid. This flag MUST be set if the data to be compressed will not fit in the history buffer at the current offset. |
| L1\_NO\_COMPRESSION  0x02 | No compression was performed. The output data consists of raw literals, and there is no match data. This occurs when the input data has no corresponding matches available in the history buffer long enough to produce a reduction in data size by RDP6.1-BC compression. The output data MUST be appended to the level-1 history buffer. |
| L1\_COMPRESSED  0x01 | Compression with the level-1 compressor was performed. The output data MUST contain at least one match. The output data MUST be appended to the level-1 history buffer. |
| L1\_INNER\_COMPRESSION  0x10 | Indicates that additional level-2 compression (using RDP 5.0 bulk compression) has been performed on the level-1 compressor output. The level-2 compression flags MUST contain the result of the chained compression. |

The level-2 compression flags are described in [MS-RDPBCGR] sections 3.1.8.2.1 and 3.1.8.3.

##### Decompressing Data

Decompression with RDP6.1-BC is based on the principles specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.1.8.3 with the compressed stream being decoded using the scheme described in section [2.2.2.4.1](#Section_4b2427d8f80048229a3d82cf1ccd3ce7) under the control of the level-1 and level-2 compression flags. For a general description of how compression chaining works during decompression, refer to section [3.1.8.1.3](#Section_2757a6b238954bd39b14decbadd89809).

| Compression Flag | Meaning |
| --- | --- |
| L1\_PACKET\_AT\_FRONT  0x04 | The level-1 history buffer MUST be reinitialized (by filling it with zeros). This flag indicates that the data to be decompressed will not fit into the local level-1 history buffer at the current offset. |
| L1\_NO\_COMPRESSION  0x02 | No compression was performed. The input data consists of raw literals that MUST be appended to the local level-1 history buffer. |
| L1\_COMPRESSED  0x01 | Compression with the level-1 compressor was performed. The input data MUST contain at least one match. The uncompressed data MUST be appended to the level-1 history buffer. |
| L1\_INNER\_COMPRESSION  0x10 | Indicates that additional level-2 compression (using RDP 5.0 bulk compression) has been performed on the level-1 compressor output. If the L1\_INNER\_COMPRESSION flag is set, the data following the Level2ComprFlags field in the RDP 6.1 Compressed Data structure (see sections 2.2.2.4.1 and [2.2.2.5.1](#Section_9b422f698e054c6db6fbfa02ef75a8f2)) MUST first be passed to the chained level-2 RDP 5.0 bulk decompressor. The level-2 decompression MUST be controlled by the contents of the Level2ComprFlags field. After this decompression phase the output MUST then be processed by the RDP6.1-BC decompressor using the Level1ComprFlags field to control the decompression. |

The following flowchart describes the how the RDP6.1-BC decompression algorithm operates.

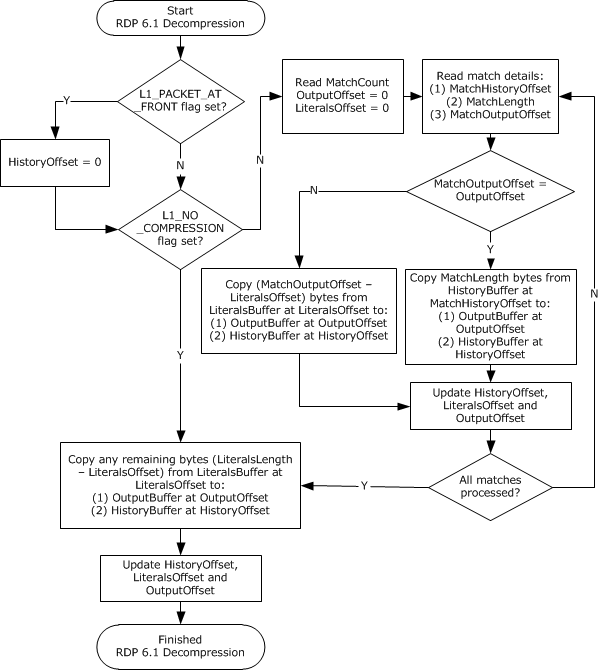


Figure : The RDP 6.1 bulk decompression algorithm

###### Data Decompression Example

To clarify the decompression principles described in section [3.1.8.2.3](#Section_c33884dec58c43b883261f93076a0504), consider decompressing the stream of data that was generated in the compression example in section [3.1.8.2.2.1](#Section_8aa890d901dd4f7e859dd751056ebe93). Note that in this example, level-2 decompression has already been applied and the input data is for the level-1 decompressor.

The data arrives in two packets. Input Data A appears as follows.

1. a b c d e f g h i j

Input Data B appears as follows.

1. [2 matches]
2. [MatchHistoryOffset@3, MatchLength=9, MatchOutputOffset@5]
3. [MatchHistoryOffset@0, MatchLength=4, MatchOutputOffset@14]
4. k l m n o u

Input Data A contains the L1\_NO\_COMPRESSION flag, as it does not contain any matches. The decompressor adds this data to its local level-1 history buffer, resulting in the following history buffer (HistoryOffset = 10).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j

The output buffer is then updated.

1. a b c d e f g h i j

Now, Input Data B contains the L1-COMPRESSED flag because it contains history matches (in this case it contains two matches). To handle this case correctly, the decompressor uses two additional offsets (in addition to the HistoryOffset). The first is the OutputOffset which is the offset in the current decompressed output. The second is the LiteralsOffset which points to the current offset in the literals section of the received data. Both of these offsets are initialized to zero before processing Input Data B.

Next, the decompressor examines the first match description. This contains MatchOutputOffset@5. Because the OutputOffset is currently zero (and not 5), this means that this match is preceded by the difference of the two offsets in raw literals (in this case, 5 literals). Hence, the decompressor copies 5 literals from the literals section of the received data to the history and output buffers, and then updates all three offsets by 5. The output buffer appears as follows (OutputOffset = 5).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. k l m n o

The history buffer appears as follows (HistoryOffset = 15).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j k l m n o

Now that the OutputOffset and MatchOutputOffset are the same (both equal to 5), we process the first match (copy-tuple). The match details instructs the decompressor to copy 9 bytes starting at history offset 3 to the history and output buffers. This action is followed by updating the OutputOffset and HistoryOffset. The output buffer appears as follows (OutputOffset = 14).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. k l m n o d e f g h i j k l

The history buffer appears as follows (HistoryOffset = 24).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j k l m n o d e f g h i j k l

The next match (MatchHistoryOffset@0, MatchLength=4, MatchOutputOffset@14) has the same MatchOutputOffset value as the OutputOffset, so there are no raw literals to process, only 4 bytes to copy to the output and history buffers, and the output and history offsets to update. The output buffer appears as follows (OutputOffset = 18).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. k l m n o d e f g h i j k l a b c d

The history buffer appears as follows (HistoryOffset = 28).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j k l m n o d e f g h i j k l a b c d

Finally, there are no matches left to process. However, some literal raw data is left in the literal buffer (the literal length of 6 is greater than the LiteralsOffset value of 5). Hence, the remaining bytes are copied to the history and output buffers (in this case, it is only 1 byte). The final output buffer appears as follows (OutputOffset = 19).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. k l m n o d e f g h i j k l a b c d u

The final history buffer appears as follows (HistoryOffset = 29).

1. 0 1 2 3
2. 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
3. a b c d e f g h i j k l m n o d e f g h i j k l a b c d u

### RDP 6.0 Bitmap Compression

RDP 6.0 Bitmap Compression is used when the RDP session color depth is 32 bits per pixel (bpp) and the bitmap of interest is either 24 bpp (RGB with no alpha channel) or 32 bpp (RGB with an alpha channel). The capability of a server to encode and a client to decode with RDP 6.0 Bitmap Compression is advertised in the Bitmap Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.2).

The main focus of RDP 6.0 Bitmap Compression is not to compress the bitmap as a single opaque block of binary data. Instead it manipulates each of the color layers as separate planes and attempts to compress them individually using a collection of compression techniques. These techniques attempt to leverage any redundancy in the color representation of the bitmaps and can additionally be followed by a simple scan-line run-length compressor if it can further compress the planes.

#### Bitmap Compression Techniques

##### Splitting and Combining Color Planes

Planar separation is a bitmap representation in which each color plane in a bitmap is sent as separate 8-bpp bitmaps. The color schemes used in RDP 6.0 Bitmap Compression are ARGB, [**RGB**](#gt_2c716d3a-e60b-4e52-bbb0-2fdeb298003b), AYCoCg, and [**YCoCg**](#gt_fb842fec-8cf7-4b40-828e-b47b505e27f7). These schemes have the following color planes.

| Color scheme | Color planes |
| --- | --- |
| ARGB | Alpha, Red, Green, Blue |
| RGB | Red, Green, Blue |
| AYCoCg | Alpha, Luma, Orange Chroma, Green Chroma |
| YCoCg | Luma, Orange Chroma, Green Chroma |

Splitting a bitmap into constituent color planes is a straightforward operation that simply involves the transfer of data into the correct color plane. For example, the 3-pixel ARGB bitmap (A1R1G1B1; A2R2G2B2; A3R3G3B3) can be split into the four color planes (A1A2A3; R1R2R3; G1G2G3; B1B2B3). In the case of 24-bpp bitmaps (or fully opaque 32-bpp bitmaps), the alpha mask for each pixel is assumed to be 0xFF, and hence, the alpha plane MAY be excluded as it is trivial to reconstitute (see the DRAW\_ALLOW\_SKIP\_ALPHA flag in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

The combination of separate color planes into a single 24-bpp or 32-bpp bitmap is also a trivial operation and merely involves reconstructing the individual ARGB or AYCoCg pixels from the constituent color planes so that the component bytes are ordered correctly. For example, the four AYCoCg planes (A1A2A3; Y1Y2Y3; Co1Co2Co3; Cg1Cg2Cg3) can be reconstituted to produce the 3-pixel (A1Y1Co1Cg1; A2Y2Co2Cg2; A3Y3Co3Cg3) bitmap.

Note that Microsoft RDP servers incorrectly convert 24 bpp RGB bitmaps to the YCoCg color space when color loss reduction (see the DRAW\_ALLOW\_DYNAMIC\_COLOR\_FIDELITY (0x02) flag in [MS-RDPBCGR] section 2.2.7.1.2) is in effect. This results in an incorrect value for the Co component. To compensate for this issue, client decoders MUST swap the B and R components after applying the inverse transformation, which converts from the YCoCg color space to the RGB color space.

##### Color Space Conversion

Color channel-based compression does not operate on the ARGB color space but rather a color space based on Luminosity (Y) and differentials of two color components: Orange (Co) and Green (Cg). Conversion between the ARGB and AYCoCg color spaces can be performed by using forward and inverse transformations (represented using matrix multiplication). The Alpha channel (A) is never converted in either direction.

The forward transformation to convert from ARGB to AYCoCg is as follows.

1. A = A
2. |Y| | 1/4 1/2 1/4| |R|
3. |Co| = | 1 0 -1 | \* |G|
4. |Cg| |-1/2 1 -1/2| |B|

The inverse transformation to convert from AYCoCg to ARGB is as follows.

1. A = A
2. |R| |1 1/2 -1/2| |Y |
3. |G| = |1 0 1/2| \* |Co|
4. |B| |1 -1/2 -1/2| |Cg|

Note that Microsoft RDP servers incorrectly convert 24 bpp RGB bitmaps to the YCoCg color space when color loss reduction (see the DRAW\_ALLOW\_DYNAMIC\_COLOR\_FIDELITY (0x02) flag in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.2) is in effect. This results in an incorrect value for the Co component. To compensate for this issue, client decoders MUST swap the B and R components after applying the inverse transformation, which converts from the YCoCg color space to the RGB color space.

The allowed ranges of the individual color planes and their lengths in bits are as follows.

| Color plane | Range | Type |
| --- | --- | --- |
| Alpha (A) | 0 (transparent) – 255 (opaque) | 8 bits |
| Red (R) | 0 – 255 | 8 bits |
| Green (G) | 0 – 255 | 8 bits |
| Blue (B) | 0 – 255 | 8 bits |
| Luma (Y) | 0 – 255 | 8 bits |
| Orange Chroma (Co) | -255 – 255 | 9 bits (two's complement) |
| Green Chroma (Cg) | -255 – 255 | 9 bits (two's complement) |

When performing the color space conversion, the data types used for all calculations MUST be at least 9 bits long to accommodate the orange and green chroma channels. In the inverse transformation, after the red, green and blue values have been calculated with 9-bit precision, they MUST all be converted to 8-bit precision in order to be stored in the decoded image. This calculation MUST be performed by taking the closest value within the 0 to 255 range (this technique is referred to as "clamping between 0 and 255"). For example:

1. -14 clamped = 0
2. 123 clamped = 123
3. 254 clamped = 254
4. 300 clamped = 255
5. 421 clamped = 255

##### Chroma Subsampling and Super-Sampling

Chroma subsampling is a compression technique that employs a lower resolution for representing the chroma planes (Co and Cg) while keeping the luminosity plane (Y) at full resolution. This method relies on the human eye being less sensitive to color than differences in luminosity. Support for chroma subsampling is advertised in the Bitmap Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

The subsampling algorithm used employs a simple averaging of four neighboring pixels in a 2-by-2 grid. This reduces the information needed to be sent by half in each dimension, resulting in a four-fold decrease for a two-dimensional bitmap per chroma channel. If an odd number of rows or columns is present, the subsampling is performed on a 2-by-1 or 1-by-2 grid, or not at all for a corner when the bitmap has both an odd number of rows and columns.

Super-sampling of a compressed chroma plane simply results in the expansion of a subsampled pixel into a 2-by-2 grid of pixels. Because this expansion results in an area with the same dimensions as the original bitmap, odd row and column counts imply that edge pixels will be expanded into either a 2-by-1, 1-by-2 or 1-by-1 grid of pixels.

The following figure shows averaging (first stage) and final subsampling (second stage) being applied to a chroma plane of a 3-by-3 bitmap as well as the super-sampling required to reconstruct the chroma plane.

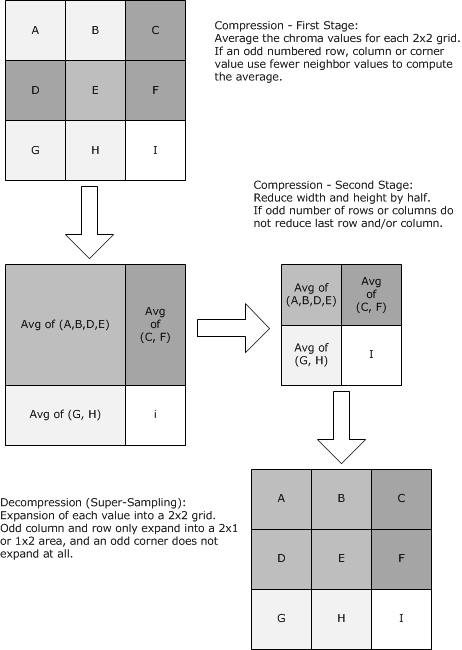


Figure : Chroma subsampling and super-sampling

##### Color Loss Reduction

Color loss reduction is a form of compression that reduces the fidelity of chroma values while maintaining the overall relative magnitude of possible chroma values and reducing the number of bits needed to represent each value. The dynamic range of the chroma representation is not reduced. This compression technique has the side effect of reducing many similar chroma values to the same reduced value, which has the potential of improving subsequent run-length compression.

The operation to reduce chroma is simply to bit shift the chroma values toward zero while adding zeros at the high-order bits. The number of bits shifted is implementation dependent and known as the Color Loss Level (CLL). The server MUST choose a value between 1 and 7 for the CLL. Usage of color loss reduction is specified in the Bitmap Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

The reverse operation to recover chroma can be performed by shifting the reduced values back toward the high-order bit and inserting zeros at the low-order bit location. The client MAY choose to perform the reverse operation using other schemes, such as a linear gradient curve, as long as the final chroma values are within the ranges specified in section [3.1.9.1.2](#Section_ef530a0a03d8482f989b57a1036797b2).

#### Run-Length Encoding

Each color plane can be individually run-length compressed. The RDP 6.0 Bitmap Compressor uses a simple scan-line compressor that breaks each scan-line into segments consisting of RAW and RUN components.

A RAW component is a non-repeating sequence in the segment, and a RUN is a sequence of the last RAW value repeated run-length times. A RUN in a segment MUST be at least three values long (shorter sequences are encoded as RAW values). In the majority of cases, a segment will contain both a RAW and a RUN component. However, depending on the scan-line to be compressed, it is possible that only a RAW or RUN component is present. In the absence of a RAW component, the RAW value used to decode the RUN component is assumed to be the value zero.

For example, an initial scan-line containing the following 12 [**ANSI characters**](#gt_372d5ad1-677d-4f38-ad65-a5849f11215f):

1. AAAABBCCCCCD

Would become the following:

1. RAW [A]; RUN [3] - Segment 1
2. RAW [BBC]; RUN [4] - Segment 2 (BB is too short to be a RUN.)
3. RAW [D]; RUN [0] - Segment 3 (The scan line is completed.)

The encoded RAW and RUN values are either absolute values (this is the case for the first scan-line) or delta values (this is the case for all other scan-lines). Delta values are relative to the previous scan-line values. The delta value calculation is performed by subtracting the previous scan-line value at the current column location from the current scan-line value being encoded.

An example of delta value usage follows (decimal values used for clarity).

1. 10, 20, 30, 40, 50, 60 - First scan-line
2. 5, 15, 25, 35, 45, 55 - Second scan-line
3. 5, 15, 25, 35, 45, 55 - Third scan-line

Converted to delta values:

1. 10, 20, 30, 40, 50, 60 - First scan-line (absolute values)
2. -5, -5, -5, -5, -5, -5 - Second scan-line (delta values)
3. 0, 0, 0, 0, 0, 0 - Third scan-line (delta values)

Which, converted to segments, become the following:

1. RAW [10,20,30,40,50,60]; RUN [0]
2. RAW [-5]; RUN [5]
3. RAW [<none>]; RUN [6] (Previous base value assumed to be 0.)

Due to the fact that the lengths of RAW and RUN components are limited to 4-bit values (see section [2.2.2.5.1.2](#Section_f7e4c717669e4f318b34e8f5ab2e107e)), individual segments may be broken up further into subsegments during encoding to produce RUN sequences consisting of more than 16 values.

##### Encoding Run-Length Sequences

Once a run-length encoder has broken up a scan-line into segments, it MUST encode each segment as one or more subsegments. The structure of segments and subsegments is the same and is defined in section [2.2.2.5.1.2](#Section_f7e4c717669e4f318b34e8f5ab2e107e).

The process of encoding scan-lines using RDP 6.0 RLE is best illustrated with a practical example. Assume that the following three scan-lines are present in a bitmap.

1. 255, 255, 255, 255, 254, 253 - First scan-line
2. 254, 192, 132, 96, 75, 25 - Second scan-line
3. 253, 140, 62, 14, 135, 193 - Third scan-line

Converted to delta values:

1. 255, 255, 255, 255, 254, 253 - First scan-line (absolute values)
2. -1, -63, -123, -159, -179, -228 - Second scan-line (delta values)
3. -1, -52, -70, -82, 60, 168 - Third scan-line (delta values)

Each value MUST be interpreted as a 1-byte two's complement signed value as follows.

1. -1, -1, -1, -1, -2, -3 - First scan-line (absolute values)
2. -1, -63, -123, 97, 77, 28 - Second scan-line (delta values)
3. -1, -52, -70, -82, 60, -88 - Third scan-line (delta values)

The next step is to encode the delta values. If the delta value does not have the most significant bit set (0 – 127), shift it 1 bit toward the highest bit. If the delta value has the most significant bit set (-1 to -128), take the 8-bit two's complement of the value (invert the bits and add one or subtract from 256), bit shift it toward the high bit value (multiply by 2), and decrement by 1. This transformation results in the following.

1. -1, -1, -1, -1, -2, -3 - First scan-line (absolute values)
2. 1, 125, 11, -62, -102, 56 - Second scan-line (delta values)
3. 1, 103, -117, -93, 120, -81 - Third scan-line (delta values)

The following represent these signed two's complement values as 8-bit hexadecimal values:

1. 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFD - First scan-line
2. 0x01, 0x7D, 0xF5, 0xC2, 0x9A, 0x38 - Second scan-line
3. 0x01, 0x67, 0x8B, 0xA3, 0x78, 0xAF - Third scan-line

Which, converted to RLE segments, become the following:

1. RAW [0xFF]; RUN [3] - Segment 1
2. RAW [0xFE,0xFD]; RUN [0] - Segment 2
3. RAW [0x01,0x7D,0xF5,0xC2,0x9A,0x38]; RUN [0] - Segment 3
4. RAW [0x01,0x67,0x8B,0xA3,0x78,0xAF]; RUN [0] - Segment 4

Using the structure defined in section 2.2.2.5.1.2, the first segment, which uses absolute values (see section [3.1.9.2](#Section_c5f101a07e674f3ca678d9760ce8921c)), is encoded as follows.

1. controlByte: (cRawBytes = 1, nRunLength = 3); rawValues: 0xFF

Which yields the following byte stream for the first segment:

1. 0x13, 0xFF - Segment 1

Encoding the remaining segments yields the following:

1. 0x20, 0xFE, 0xFD - Segment 2
2. 0x60, 0x01, 0x7D, 0xF5, 0xC2, 0x9A, 0x38 - Segment 3
3. 0x60, 0x01, 0x67, 0x8B, 0xA3, 0x78, 0xAF - Segment 4

The following flow chart illustrates how a given set of RAW values and a RUN (which can be zero) is encoded using RDP 6.0 RLE.

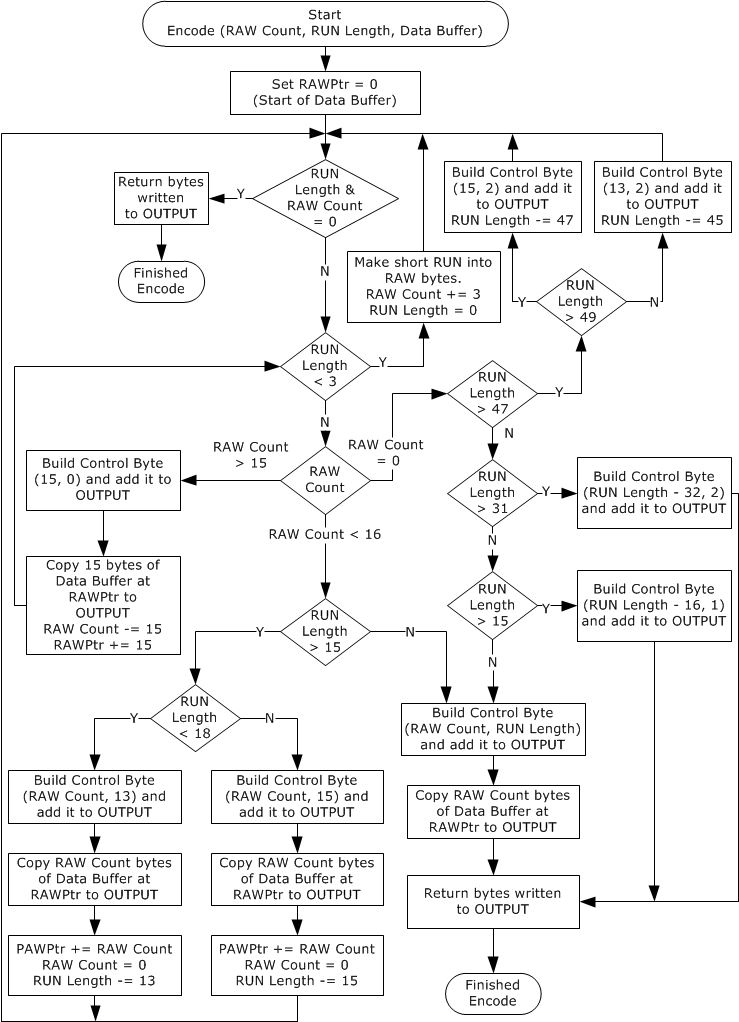


Figure : Encoding data using RDP 6.0 Run-Length Encoding (RLE)

##### Extra Long RUN Sequences

The lengths of RAW and RUN components are limited to 4-bit values and, as a result, encoding of run-lengths greater than 15 values requires special casing, as described in section [2.2.2.5.1.2](#Section_f7e4c717669e4f318b34e8f5ab2e107e).

Encoding of an extra long RUN sequence is best described with an example. Assume that there is a pattern consisting of the letter A repeated 100 times. The resulting RLE segments are the following:

1. RAW [A]; RUN [15] - Segment 1
2. RAW [<none>]; RUN [47] - Segment 2
3. RAW [<none>]; RUN [37] - Segment 3

Using the structure and rules in section 2.2.2.5.1.2, these segments are encoded as the following:

1. controlByte: (1, 15); rawValues: A
2. controlByte: (15, 2); rawValues: <none>
3. controlByte: (5, 2); rawValues: <none>

In hexadecimal, this becomes the following:

1. 0x1F, 0x41, 0xF2, 0x52

##### Decoding Run-Length Sequences

Encoding of run-length sequences ensures that there is at least one subsegment per scan-line. The control byte described in section [2.2.2.5.1.2](#Section_f7e4c717669e4f318b34e8f5ab2e107e) contains all of the information necessary to decode the sequence bytes.

The process of decoding an encoded sequence of bytes for a color plane using RDP 6.0 RLE is best illustrated with a practical example. Assume that the following bytes are from a color plane in a bitmap that is 6 pixels wide by 3 pixels high.

1. 0x13, 0xFF, 0x20, 0xFE, 0xFD, 0x60, 0x01, 0x7D,
2. 0xF5, 0xC2, 0x9A, 0x38, 0x60, 0x01, 0x67, 0x8B,
3. 0xA3, 0x78, 0xAF

The first byte is the control byte of the first segment from the first scan-line.

1. 0x13 -> Control byte = 1 RAW value, 3-length RUN of the last raw value
2. 0xFF -> Raw value

Hence, decoding the first segment according to the information in the control byte yields the following:

1. 0xFF, 0xFF, 0xFF, 0xFF

Because these values are from the first scan-line, the values are absolute. Shown in unsigned decimal, the values are the following:

1. 255, 255, 255, 255

The second segment control byte is the following:

1. 0x20

This implies two raw values and a zero-length run, resulting in the following two unsigned decimal absolute values.

1. 0xFE, 0xFD

This completes the first scan-line, and all values decoded from this point are deltas and MUST be specially decoded. Before decoding, the deltas MUST be interpreted as unsigned 1-byte values. If the encoded delta value is odd, then decrement it by 1, shift it 1 bit toward the lowest bit, and subtract it from 255. This yields the original absolute value. If the encoded delta value is even, shift it 1 bit toward the lowest bit. This yields the original absolute value.

Examining the third segment yields six raw values and a zero-length run.

1. 0x01, 0x7D, 0xF5, 0xC2, 0x9A, 0x38

The deltas MUST be decoded. The unsigned decimal representation of the deltas is as follows.

1. 1, 125, 245, 194, 154, 56

Applying the delta transformation to the deltas in the second scan-line yields the following 1-byte unsigned values.

1. 255, 193, 133, 97, 77, 28

To compute the final absolute values for the second row, the unsigned, 8-bit delta values are added to the absolute values of the first scan-line using 1-byte arithmetic.

1. Column 1: 255 + 255 = 254
2. Column 2: 255 + 193 = 192
3. Column 3: 255 + 133 = 132
4. Column 4: 255 + 97 = 96
5. Column 5: 254 + 77 = 75
6. Column 6: 253 + 28 = 25

This completes the second scan-line, and the final absolute values are the following:

1. 254, 192, 132, 96, 75, 25

The third segment has six raw values and a zero-length run.

1. 0x01, 0x67, 0x8B, 0xA3, 0x78, 0xAF

The decoded unsigned 8-bit unsigned values representing the final deltas are as follows.

1. 255, 204, 186, 174, 60, 168

To compute the final absolute values for the third row, the unsigned 8-bit delta values are added to the absolute values of the second scan-line using 1-byte arithmetic.

1. Column 1: 254 + 255 = 253
2. Column 2: 192 + 204 = 140
3. Column 3: 132 + 186 = 62
4. Column 4: 96 + 174 = 14
5. Column 5: 75 + 60 = 135
6. Column 6: 25 + 168 = 193

This completes the second scan-line, and the final absolute values are the following:

1. 253, 140, 62, 14, 135, 193

The fully decoded plane is the following:

1. 255, 255, 255, 255, 254, 253
2. 254, 192, 132, 96, 75, 25
3. 253, 140, 62, 14, 135, 193

#### Compressing a Bitmap

The overall scheme used to compress a bitmap with RDP 6.0 Bitmap Compression is described in the following figure. This figure effectively shows how the techniques described in section [3.1.9](#Section_0d2e1a03e12346b2b2b8ed730a794ae4) are employed. The usage of color reduction and chroma subsampling is specified in the Bitmap Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.2). The color loss level to use (see section [3.1.9.1.4](#Section_b4462e6422864e779c5e2046fc490cc0)) is not exchanged on the wire and is server implemention-dependent.

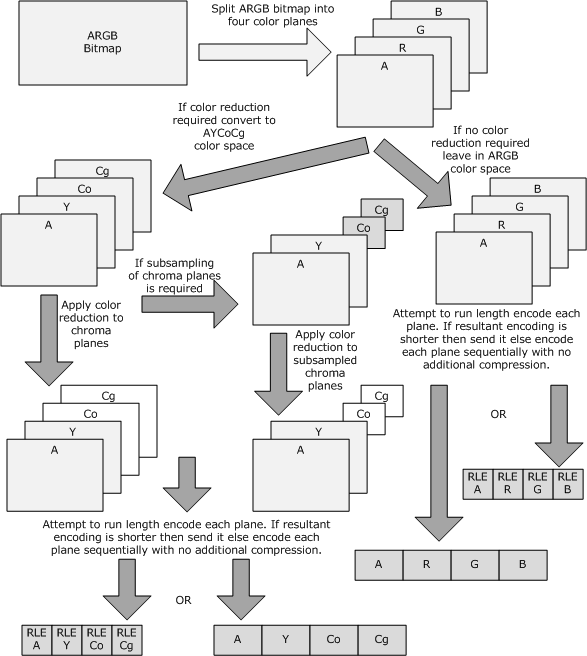


Figure : Compressing a bitmap using RDP 6.0 Bitmap Compression

#### Decompressing a Bitmap

The decision-flow used to decompress a bitmap that is compressed with RDP 6.0 Bitmap Compression is illustrated in the following figure. The flags describing the bitmap data (present in the compressed data format header) are defined in section [2.2.2.5.1](#Section_9b422f698e054c6db6fbfa02ef75a8f2).

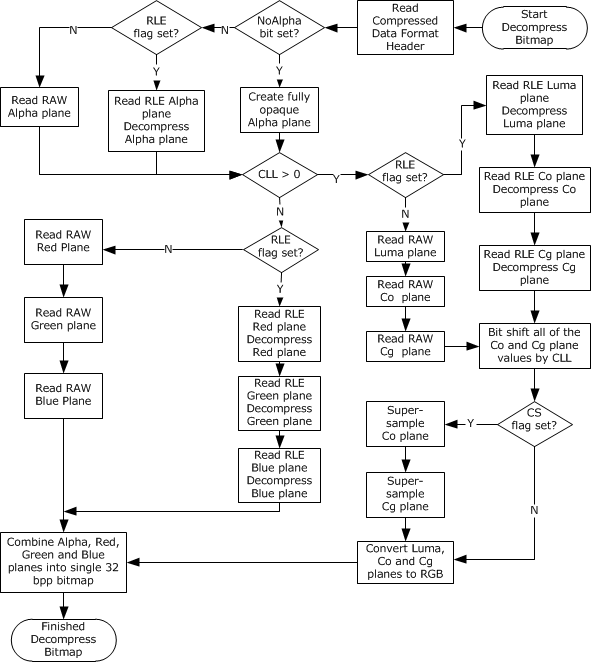


Figure : Decompression of a bitmap compressed with RDP 6.0 Bitmap Compression

## Client Details

### Abstract Data Model

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

**Note**  It is possible to implement the following conceptual data by using a variety of techniques as long as the implementation produces external behavior that is consistent with that described in this document.

#### Primary Drawing Order History

The Primary Drawing Order History store holds information on the fields that have been received in primary drawing orders. There are three pieces of information that MUST be recorded:

* Last primary order type processed.
* Current bounding rectangle.
* Per-order record of the last value received in each field.

These records are updated as each primary drawing order is processed, and are used to efficiently decode and process primary drawing orders received from the server (see section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b)).

#### Save Bitmap

The Save Bitmap is a 480-by-480-pixel bitmap that is used to temporarily store desktop bitmap images received in the SaveBitmap Primary Drawing Order (see section [3.2.5.1.1.1.12](#Section_9a2050e0e6454452848c5b82834b2a6f)). Bitmaps stored in the Save Bitmap are tiled to maximize the number of bitmaps that can be stored (a graphical representation of how bitmaps are stored is shown in the figure in section [4.5](#Section_5337b5e437084570a6821791ffc90c88)). If support for the SaveBitmap Primary Drawing Order is not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the Save Bitmap is not required.

#### Bitmap Cache

A Bitmap Cache is a store that contains bitmap images that were sent to the client by using either the Cache Bitmap – Revision 1 (section [2.2.2.2.1.2.2](#Section_C9365B320BE547C9AF1AFEBA2EA1EE9F)), Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3](#Section_3C76F8FBFCAA4CA69D2E3F449BDBED40)), or Cache Bitmap – Revision 3 (section [2.2.2.2.1.2.8](#Section_710CF5C26A0C4CEA8D62FA4F639D7C5A)) Secondary Drawing Order.

#### Persistent Bitmap Cache

A Persistent Bitmap Cache is a store that contains bitmap images that were sent to the client by using the Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40)) or Cache Bitmap - Revision 3 (section [2.2.2.2.1.2.8](#Section_710cf5c26a0c4cea8d62fa4f639d7c5a)) Secondary Drawing Order. Unlike the Bitmap Cache described in section [3.2.1.3](#Section_4a997b03fdf243e4a2a29a749ddda8cc), a Persistent Bitmap Cache is not bound to the lifetime of a given RDP connection and its contents are persisted even after the RDP connection is closed.

#### Persisted Bitmap Keys

The Persisted Bitmap Keys store holds a collection of 64-bit bitmap keys, each of which uniquely identifies a bitmap image that is present in a Persistent Bitmap Cache (section [3.2.1.4](#Section_0a09fd5178624c1a9846a5a79a02cdac)). The lifetime of this store is bound to the lifetime of the Persistent Bitmap Caches.

### Timers

None.

### Initialization

None.

### Higher-Layer Triggered Events

None.

### Processing Events and Sequencing Rules

#### Drawing Orders

All drawing orders are encapsulated in an [Orders Update (section 2.2.2.1)](#Section_ef7c27f5097746fea50ef9a51d6ab90b), which is received as part of the Graphics Update PDU (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.1.3); or they are encapsulated in a [Fast-Path Orders Update (section 2.2.2.2)](#Section_7f53bb2892e94cbea15f8553c902e3c4), which is received as part of the Fast-Path Update PDU (see [MS-RDPBCGR] section 2.2.9.1.2.1).

There are three classes of drawing orders:

* [Primary Drawing Orders (section 2.2.2.2.1.1)](#Section_b0d203b0b32d4b0d84e5bc21eb11a3ec)
* [Secondary Drawing Orders (section 2.2.2.2.1.2)](#Section_9df0ec924b494302aa863421151b0c11)
* [Alternate Secondary Drawing Orders (section 2.2.2.2.1.3)](#Section_b69a751e34df4326b4e10e582ff7ea97)

Orders belonging to each of these classes are packed together into an Orders Update structure or a Fast-Path Orders Update structure, each order being aligned on a byte boundary.

##### Primary Drawing Orders

###### Processing Primary Drawing Orders

All primary drawing orders MUST conform to the structure and rules defined in section [2.2.2.2.1.1.2](#Section_23f766d483434e6b8281071ddccc0272).

To efficiently decode and process a primary drawing order, the client MUST use a [Primary Drawing Order History (section 3.2.1.1)](#Section_f1da06b585c940c7a0de3f280f480960) store. This store holds three pieces of information:

* Last primary order type processed
* Current bounding rectangle
* Per-order record of the last value received for each field

These stored records allow the client to correctly decode a received primary drawing order, as only fields that have new values are required to be included in orders sent by the server. If a field is unchanged from the value that it had when the order type was last sent, it SHOULD NOT be included in the order. The fields that are present in the order MUST be indicated by the fieldFlags field.

If all of the Coord-type fields (see section [2.2.2.2.1.1.1.1](#Section_870b982c9abe40cb99d02c3f0cc5fb74)) in an order can be represented as a signed delta in the range -127 to 128 from the previous field value, all of these fields SHOULD contain delta-coordinates (see sections 2.2.2.2.1.1.1.1 and [2.2.2.2.1.1.1.2](#Section_008cb2338ae94e05ad0383f284f20a61)). The presence of delta-coordinates MUST be indicated by the TS\_DELTA\_COORDINATES (0x10) flag in the primary drawing header.

The basic process to decode a primary drawing order begins with reading the controlFlags header byte (this is common to all drawing classes) to determine the order class. If order is a primary drawing order, the following sequence of steps SHOULD be used to decode the order:

1. The control flags extract the following information:
   * Whether or not the order type is specified
   * The fields that are present in the primary order field data
   * Whether or not any components of the bounding rectangle are present
   * Whether or not delta-coordinates are used in the primary order Coord-type fields
2. If support for the primary drawing order was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the client SHOULD ignore the order, and processing SHOULD cease.
3. Determine whether or not all of the data required to decode and process the order has been received in the [Orders Update (section 2.2.2.1)](#Section_ef7c27f5097746fea50ef9a51d6ab90b) structure or the [Fast-Path Orders Update (section 2.2.2.2)](#Section_7f53bb2892e94cbea15f8553c902e3c4) structure. If the packet does not contain enough data, processing SHOULD cease, and the order update SHOULD be ignored.
4. Read the order data, and validate the fields to make sure all the field data is consistent with the order specification (for example, the maximum number of fields and maximum order size MUST conform to the order specification). If any of the field data for a given order is inconsistent or refers to non-existent or invalid items (such as a non-existent cache entry or invalid brush format), processing of the order SHOULD terminate, and it SHOULD also be ignored.

Once the order has been decoded, and all of the information necessary to process it has been collected, the data MUST be handed off to a graphics rendering module so that the images from the remote system can be displayed locally on the client system. The client MUST also update the records in the [Primary Drawing Order History (section 3.3.1.2)](#Section_19b1b7f2456c4d928f5a5f507947889c) to ensure that future orders can be decoded correctly.

Processing of DstBlt

The structure and fields of the DstBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.1](#Section_87ea30df59d6438ea73583f0225fbf91), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of MultiDstBlt

The structure and fields of the MultiDstBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.2](#Section_aaae78403eb440389204b95f87b44534), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of PatBlt

The structure and fields of the PatBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.3](#Section_bd4bf5e7b98845f982013b22cc9aeeb8), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If a cached brush is specified in this order, that brush MUST have been received by the client in a prior Cache Brush Secondary Drawing Order (see section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)). If this is not the case, the client SHOULD ignore this order. Furthermore, if support for brush caching was not specified in the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7), and a cached brush is included in the PatBlt order, the client SHOULD ignore this order.

Processing of MultiPatBlt

The structure and fields of the MultiPatBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.4](#Section_5f677576a62e48d9a58d711490882365), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If a cached brush is specified in this order, that brush MUST have been received by the client in a prior Cache Brush Secondary Drawing Order (see section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)). If this is not the case, the client SHOULD ignore this order. Furthermore, if support for brush caching was not specified in the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7), and a cached brush is included in the MultiPatBlt order, the client SHOULD ignore this order.

Processing of OpaqueRect

The structure and fields of the OpaqueRect Primary Drawing Order are specified in section [2.2.2.2.1.1.2.5](#Section_1eead7aaac63411a9f8cb1b227526877), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of MultiOpaqueRect

The structure and fields of the MultiOpaqueRect Primary Drawing Order are specified in section [2.2.2.2.1.1.2.6](#Section_74e46cd9d6494bbe9f8af4200f23302e), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of ScrBlt

The structure and fields of the ScrBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.7](#Section_a4e322b0cd644dfc8e1af24dc0edc99d), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of MultiScrBlt

The structure and fields of the MultiScrBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.8](#Section_31b79b4915b74709bbc2d58ce0362292), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of MemBlt

The structure and fields of the MemBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.9](#Section_84c2ec2ff776405b9b486894a28b1b14), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

The source bitmap associated with the MemBlt order MUST reside in one of two possible locations:

1. One of the bitmap caches (section [3.1.1.1.1](#Section_2bf9258842bd45278b3eb90c56e292d2))
2. The Offscreen Bitmap Cache (section [3.1.1.1.5](#Section_63061daa980843c9b00dd5eda7dc8c21))

If the bitmap cache ID (specified as part of the **cacheId** field) refers to one of the specified bitmap caches, the actual bitmap data MUST have been received in a Cache Bitmap Secondary Drawing Order (sections [3.2.5.1.2.1.1](#Section_95b802fc02f643e5a60e970e77b70872), [3.2.5.1.2.1.2](#Section_88a45af10a9c4cecb1c2dd87829dbc1f), and [3.3.5.1.2.1.7](#Section_cb71c3b78acd48aca13fe7900e202ef3)). If the **cacheIndex** field is set to BITMAPCACHE\_WAITING\_LIST\_INDEX (32767), the last bitmap cache entry MUST contain the source bitmap; this implies that the bitmap data was received in a Cache Bitmap (Revision 2) Order, and the CBR2\_DO\_NOT\_CACHE (0x10) flag was set or it was received in a Cache Bitmap (Revision 3) Order, and the CBR3\_DO\_NOT\_CACHE (0x10) flag was set.

However, if the bitmap cache ID is set to TS\_BITMAPCACHE\_SCREEN\_ID (0xFF), the source bitmap resides in the Offscreen Bitmap Cache at the entry location specified by the **cacheIndex** field. This entry MUST have been created and initialized in response to the processing of prior Create Offscreen Bitmap (see section [3.2.5.1.3.1.1](#Section_8594bbe257214cc4bf0ef03419acfa5d)) and Switch Surface (see section [3.2.5.1.3.1.2](#Section_434647bd7aba48fcb96ac1a7b433feb4)) Alternate Secondary Drawing Orders; the Create Offscreen Bitmap Order creates the offscreen bitmap while the Switch Surface Order is used to redirect all drawing operations to the offscreen bitmap.

Processing of Mem3Blt

The structure and fields of the Mem3Blt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.10](#Section_3aa21a4a031c4a3f8fc5159068f30237), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

The decoding and processing of the Mem3Blt Order follow the same principles as those outlined for the MemBlt Order specified in section [3.2.5.1.1.1.9](#Section_5bf823d9eeb74cfcb7a8dc1037d2426a). However, the Mem3Blt Order includes fields to specify a brush.

If a cached brush is specified in this order, that brush MUST have been received by the client in a prior Cache Brush Secondary Drawing Order (see section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)). If this is not the case, the client SHOULD ignore this order. Furthermore, if support for brush caching was not specified in the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), and a cached brush is included in the Mem3Blt Order, the client SHOULD ignore this order.

Processing of LineTo

The structure and fields of the LineTo Primary Drawing Order are specified in section [2.2.2.2.1.1.2.11](#Section_27276fcf32be49a99a1797ec6167ac7e), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of SaveBitmap

The structure and fields of the SaveBitmap Primary Drawing Order are specified in section [2.2.2.2.1.1.2.12](#Section_db4ce8dbcf76483d99ba354cd010ea4a), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order. Bitmap data received in this order MUST be stored in the Save Bitmap (see section [3.2.1.2](#Section_c464f205346c475fb3d228fcef86a5ad)).

Processing of GlyphIndex

The structure and fields of the GlyphIndex Primary Drawing Order are specified in section [2.2.2.2.1.2.5](#Section_6dc06088b1124f4fb0bcb97bba621a0c), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If support for glyph caching was not specified in the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8), the client SHOULD ignore this order as it requires the existence of the glyph caches (see section [3.1.1.1.2](#Section_6d4a8cc0d4814766b609778b09c1ceac)).

All of the glyphs associated with the glyph cache indices specified in the order MUST have been received by the client in a prior Revision 1 or 2 Cache Glyph Secondary Drawing Order (see sections [3.2.5.1.2.1.4](#Section_38c3adff5fa94eddace958de1bdd515e) and [3.2.5.1.2.1.5](#Section_74253730bb694905b21be7324a083f38)), or a FastGlyph Primary Drawing Order (see section [3.2.5.1.1.1.15](#Section_0f49171ed4934d998a963b1d9ae11bab)). If a fragment cache index is specified in a USE clause, that fragment MUST have been cached while processing an ADD clause in a prior GlyphIndex or FastIndex Order.

If a cached brush is specified in this order, that brush MUST have been received by the client in a prior Cache Brush Secondary Drawing Order (see section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)). If this is not the case, the client SHOULD ignore this order. Furthermore, if support for brush caching was not specified in the Brush Capability Set (see [MS-RDPBCGR] section 2.2.7.1.7), and a cached brush is included in the GlyphIndex order, the client SHOULD ignore this order.

Processing of FastIndex

The structure and fields of the FastIndex Primary Drawing Order are specified in section [2.2.2.2.1.1.2.14](#Section_ac55f919acb943c1a3ec33802929f9f7), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

The decoding and processing of the FastIndex Order follow the same principles as those outlined for the GlyphIndex Order specified in section [3.2.5.1.1.1.13](#Section_e30f1b8cdd85420f8bde1376e03dae4b). However, the FastIndex Order does not use cached brushes and also utilizes a more efficient field encoding.

Processing of FastGlyph

The structure and fields of the FastGlyph Primary Drawing Order are specified in section [2.2.2.2.1.1.2.15](#Section_839e963ce24a42adb4bdeb108340629c), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If support for glyph caching was not specified in the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8), the client SHOULD ignore this order because it requires the existence of the glyph caches (see section [3.1.1.1.2](#Section_6d4a8cc0d4814766b609778b09c1ceac)).

All of the glyphs associated with the glyph cache indices specified in the order MUST have been received by the client in a prior Revision 1 or 2 Cache Glyph Secondary Drawing Order (see sections [3.2.5.1.2.1.4](#Section_38c3adff5fa94eddace958de1bdd515e) and [3.2.5.1.2.1.5](#Section_74253730bb694905b21be7324a083f38)) or a FastGlyph Primary Drawing Order.

Once the client has completed decoding and processing the FastGlyph order, and the glyph has been rendered successfully, the glyph data MUST be stored in the specified glyph cache.

Processing of PolygonSC

The structure and fields of the PolygonSC Primary Drawing Order are specified in section [2.2.2.2.1.1.2.16](#Section_22e1117fd40d41df970959389ac1f8dc), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of PolygonCB

The structure and fields of the PolygonCB Primary Drawing Order are specified in section [2.2.2.2.1.1.2.17](#Section_24f4e2291f9f4633b9be66d7579ba36d), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If a cached brush is specified in this order, that brush MUST have been received by the client in a prior Cache Brush Secondary Drawing Order (see section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)). If this is not the case, the client SHOULD ignore this order. Furthermore, if support for brush caching was not specified in the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7), and a cached brush is included in the PolygonCB order, the client SHOULD ignore this order.

Processing of Polyline

The structure and fields of the Polyline Primary Drawing Order are specified in section [2.2.2.2.1.1.2.18](#Section_b60b4d9baac54b9884d3edc0c0779a5c), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of EllipseSC

The structure and fields of the EllipseSC Primary Drawing Order are specified in section [2.2.2.2.1.1.2.19](#Section_5146fab3c43f44d7b54104666a80e33b), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

Processing of EllipseCB

The structure and fields of the EllipseCB Primary Drawing Order are specified in section [2.2.2.2.1.1.2.20](#Section_897203838394423a810aba631d66055f), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If a cached brush is specified in this order, that brush MUST have been received by the client in a prior Cache Brush Secondary Drawing Order (see section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)). If this is not the case, the client SHOULD ignore this order. Furthermore, if support for brush caching was not specified in the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7), and a cached brush is included in the EllipseCB order, the client SHOULD ignore this order.

Processing of DrawNineGrid

The structure and fields of the DrawNineGrid Primary Drawing Order are specified in section [2.2.2.2.1.1.2.21](#Section_44bd6a68a5b74a38980706ba4ab4a72a), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

If support for NineGrid bitmap caching was not specified in the DrawNineGrid Cache Capability Set (see section [2.2.1.2](#Section_c7fff28863db4521bbe577e060fb0780)), the client SHOULD ignore the order since the order refers to NineGrid bitmaps in the NineGrid Bitmap Cache (see section [3.1.1.1.6](#Section_b71d528445c146e7b92511edd3b1de82)).

The source bitmap (which resides in the NineGrid Bitmap Cache) MUST have been created and initialized as a result of processing prior Create NineGrid Bitmap (see section [3.2.5.1.3.1.3](#Section_30d2592f86624803961f379e7ab22c46)) and Stream Bitmap (see section [3.2.5.1.3.1.4](#Section_75377a2d99b6461f8520fc4adff29a6c)) Alternate Secondary Orders.

Processing of MultiDrawNineGrid

The structure and fields of the MultiDrawNineGrid Primary Drawing Order are specified in section [2.2.2.2.1.1.2.22](#Section_101fecdc3ad841c6b7b670d8efd4ad70), and the techniques described in section [3.2.5.1.1.1](#Section_ac9fa6b7081341bfa9c2b117c6e1bd6b) demonstrate how to decode and process the order.

The decoding and processing of the MultiDrawNineGrid Order follow the same principles as those outlined for the DrawNineGrid order specified in section [3.2.5.1.1.1.21](#Section_73caaa7677904cfa8bbe394106523a4a). However, the MultiDrawNineGrid includes multiple clipping rectangles, as opposed to the DrawNineGrid order, which only includes one clipping rectangle.

##### Secondary Drawing Orders

###### Processing Secondary Drawing Orders

All secondary drawing orders are identified by the [Secondary Drawing Order Header (section 2.2.2.2.1.2.1.1)](#Section_c54e2667335d4a59a6e27dc9744dbe79) and are used to manipulate the RDP caches (section [3.1.1.1](#Section_b1c96aad5c7245969324619a0a6a31ae)):

* The Cache Bitmap (Revision 1) Secondary Drawing Order (section [3.2.5.1.2.1.1](#Section_95b802fc02f643e5a60e970e77b70872)) manages the Revision 1 Bitmap Caches (section [3.1.1.1.1](#Section_2bf9258842bd45278b3eb90c56e292d2)).
* The Cache Bitmap (Revision 2) Secondary Drawing Order (section [3.2.5.1.2.1.2](#Section_88a45af10a9c4cecb1c2dd87829dbc1f)) and Cache Bitmap (Revision 3) Secondary Drawing Order (section [2.2.2.2.1.2.8](#Section_710CF5C26A0C4CEA8D62FA4F639D7C5A)) manage the Revision 2 Bitmap Caches (section 3.1.1.1.1).
* The Cache Color Table Secondary Drawing Order (section [3.2.5.1.2.1.3](#Section_6d97015f71e743589cf80706862eaa78)) manages the Color Table Cache (section [3.1.1.1.3](#Section_b31800c6da774acc854bb954ebc1848b)).
* The Revision 1 and 2 Cache Glyph Secondary Drawing Orders (sections [3.2.5.1.2.1.4](#Section_38c3adff5fa94eddace958de1bdd515e) and [3.2.5.1.2.1.5](#Section_74253730bb694905b21be7324a083f38)) manage the Glyph Cache (section [3.1.1.1.2](#Section_6d4a8cc0d4814766b609778b09c1ceac)).
* The Cache Brush Secondary Drawing Order (section [3.2.5.1.2.1.6](#Section_bb11083dd6b745cfa3e48347d9f1c3b3)) manages the Brush Cache (section [3.1.1.1.4](#Section_a7e8809dd24343b5be0b25f8fd827f61)).

If the client has not advertised support for a particular cache type, it SHOULD ignore any secondary and primary drawing orders associated with that cache, if they are sent by the server.

Processing of Cache Bitmap (Revision 1)

The structure and fields of the Cache Bitmap (Revision 1) Secondary Drawing Order are specified in section [2.2.2.2.1.2.2](#Section_c9365b320be547c9af1afeba2ea1ee9f). The order fields MUST be processed in accordance with this description.

If the client does not support the Revision 1 bitmap caches (specified in the Revision 1 Bitmap Cache Capability Set described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.1), or the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders, this order SHOULD be ignored.

Processing of Cache Bitmap (Revision 2)

The structure and fields of the Cache Bitmap (Revision 2) Secondary Drawing Order are specified in section [2.2.2.2.1.2.3](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40). The order fields MUST be processed in accordance with this description.

The destination cache in which to store the bitmap that is encapsulated in the cache order will either be a Standard Bitmap Cache (section [3.2.1.3](#Section_4a997b03fdf243e4a2a29a749ddda8cc)) or a Persistent Bitmap Cache (section [3.2.1.4](#Section_0a09fd5178624c1a9846a5a79a02cdac)), depending on the cache structure that was specified using the Revision 2 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) sections 2.2.7.1.4.2 and 2.2.7.1.4.2.1). If the target cache is persistent, then the server MUST include a 64-bit key that uniquely identifies the bitmap (indicated by the presence of the CBR2\_PERSISTENT\_KEY\_PRESENT (0x02) flag). The client MUST save this key in the Persisted Bitmap Keys store (section [3.2.1.5](#Section_a8f3bf9e0cd74fca859643a50ae96b27)) so that the Persistent Key List PDUs can be sent as specified in [MS-RDPBCGR] section 3.2.5.3.17.

If the client does not support the Revision 2 bitmap caches (specified in the Revision 2 Bitmap Cache Capability Set described in [MS-RDPBCGR] section 2.2.7.1.4.2), or the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders, this order SHOULD be ignored.

Processing of Cache Color Table

The structure and fields of the Cache Color Table Secondary Drawing Order are specified in section [2.2.2.2.1.2.4](#Section_49fad30181714f38a68a7394baeb7b15). The order fields MUST be processed in accordance with this description.

If the client does not support the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders, this order SHOULD be ignored. Furthermore, if a client encounters any inconsistencies or errors when decoding and processing the Cache Color Table Secondary Drawing Order, the connection SHOULD be dropped.

Processing of Cache Glyph (Revision 1)

The structure and fields of the Cache Glyph (Revision 1) Secondary Drawing Order are specified in section [2.2.2.2.1.2.5](#Section_6dc06088b1124f4fb0bcb97bba621a0c). The order fields MUST be processed in accordance with this description.

If the client does not support glyph caching (specified in the Glyph Cache Capability Set described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8) or the GlyphIndex (section 2.2.2.2.1.2.5), [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7), and [FastGlyph (section 2.2.2.2.1.1.2.15)](#Section_839e963ce24a42adb4bdeb108340629c) Primary Drawing Orders, this order SHOULD be ignored. Furthermore, if a client encounters any inconsistencies or errors when decoding and processing the Cache Glyph (Revision 1) Secondary Drawing Order, the connection SHOULD be dropped.

Processing of Cache Glyph (Revision 2)

The structure and fields of the Cache Glyph (Revision 2) Secondary Drawing Order are specified in section [2.2.2.2.1.2.6](#Section_ffc80435136e442e908540ca1d27a413). The order fields MUST be processed in accordance with this description.

The decoding and processing of the Cache Glyph (Revision 2) order follows the same principles as those outlined for the Cache Glyph (Revision 1) order specified in section [3.2.5.1.2.1.4](#Section_38c3adff5fa94eddace958de1bdd515e). However, the Revision 2 order utilizes a more efficient field encoding. Furthermore, if a client encounters any inconsistencies or errors when decoding and processing the Cache Glyph (Revision 2) Secondary Drawing Order, the connection SHOULD be dropped.

Processing of Cache Brush

The structure and fields of the Cache Brush Secondary Drawing Order are specified in section [2.2.2.2.1.2.7](#Section_9e7027c4d9894cb09674e2e36c95ebdd). The order fields MUST be processed in accordance with this description.

The primary drawing orders that use cached brushes are:

* PatBlt (see section [3.2.5.1.1.1.3](#Section_6513cdd0d7df498a88e07d7cc12cac8d))
* MultiPatBlt (see section [3.2.5.1.1.1.4](#Section_02f418fb721b49c987565b48b85ed00d))
* Mem3Blt (see section [3.2.5.1.1.1.10](#Section_bbbcf2bd676b4777aeff10f861389880))
* GlyphIndex (see section [3.2.5.1.1.1.13](#Section_e30f1b8cdd85420f8bde1376e03dae4b))
* PolygonCB (see section [3.2.5.1.1.1.17](#Section_cfb6acb407ba422b942a8e51aa21c9b0))
* EllipseCB (see section [3.2.5.1.1.1.20](#Section_e5d18a66881d4569b1857807d377ccf5))

If the client does not support brush caching (specified in the Brush Capability Set described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7) or any primary drawing orders that use cached brushes, this order SHOULD be ignored. Furthermore, if a client encounters any inconsistencies or errors when decoding and processing the Cache Brush Secondary Drawing Order, the connection SHOULD be dropped.

Processing of Cache Bitmap (Revision 3)

The structure and fields of the Cache Bitmap (Revision 3) Secondary Drawing Order are specified in section [2.2.2.2.1.2.8](#Section_710CF5C26A0C4CEA8D62FA4F639D7C5A). The order fields MUST be processed in accordance with this description.

The destination cache in which to store the bitmap that is encapsulated in the cache order will either be a Standard Bitmap Cache (section [3.2.1.3](#Section_4A997B03FDF243E4A2A29A749DDDA8CC)) or a Persistent Bitmap Cache (section [3.2.1.4](#Section_0A09FD5178624C1A9846A5A79A02CDAC)), depending on the cache structure that was specified using the Revision 2 Bitmap Cache Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) sections 2.2.7.1.4.2 and 2.2.7.1.4.2.1). The server MUST include a 64-bit key that uniquely identifies the bitmap in the Cache Bitmap (Revision 3) Order. If the target cache is persistent, the client MUST save this key in the Persisted Bitmap Keys store (section [3.2.1.5](#Section_A8F3BF9E0CD74FCA859643A50AE96B27)) so that the Persistent Key List PDUs can be sent as specified in [MS-RDPBCGR] section 3.2.5.3.17.

If the client does not support the Cache Bitmap (Revision 3) Order (support is specified by including the ORDERFLAGS\_EX\_CACHE\_BITMAP\_REV3\_SUPPORT (0x0002) flag in the **orderSupportExFlags** field of the Order Capability Set defined in [MS-RDPBCGR] section 2.2.7.1.3), then this order SHOULD be ignored. Furthermore, if the client does not support the Revision 2 bitmap caches (specified in the Revision 2 Bitmap Cache Capability Set described in [MS-RDPBCGR] section 2.2.7.1.4.2), or the MemBlt (section [2.2.2.2.1.1.2.9](#Section_84C2EC2FF776405B9B486894A28B1B14)) and Mem3Blt (section [2.2.2.2.1.1.2.10](#Section_3AA21A4A031C4A3F8FC5159068F30237)) Primary Drawing Orders, this order SHOULD be ignored.

##### Alternate Secondary Drawing Orders

###### Processing Alternate Secondary Drawing Orders

All alternate secondary drawing orders are identified by the [Alternate Secondary Drawing Order Header (section 2.2.2.2.1.3.1.1)](#Section_b93614406e084a0697e76e1b7a4cc204) and are used to manage offscreen and NineGrid bitmaps and to transport opaque GDI+ 1.1 records:

* The Create Offscreen Bitmap (see section [3.2.5.1.3.1.1](#Section_8594bbe257214cc4bf0ef03419acfa5d)) and Switch Surface (see section [3.2.5.1.3.1.2](#Section_434647bd7aba48fcb96ac1a7b433feb4)) Alternate Secondary Drawing Orders are used to manipulate the bitmaps in the Offscreen Bitmap Cache (see section [3.1.1.1.5](#Section_63061daa980843c9b00dd5eda7dc8c21)).
* The Create NineGrid Bitmap (see section [3.2.5.1.3.1.3](#Section_30d2592f86624803961f379e7ab22c46)) and Stream Bitmap (see section [3.2.5.1.3.1.4](#Section_75377a2d99b6461f8520fc4adff29a6c)) Alternate Secondary Drawing Orders are used to manipulate the bitmaps in the NineGrid Bitmap Cache (see section [3.1.1.1.6](#Section_b71d528445c146e7b92511edd3b1de82)).
* The GDI+ Alternate Secondary Drawing Orders (see section [2.2.2.2.1.3.6](#Section_095f82214abb4e6b9b38b50ceaa4a34e)) are used to transport opaque GDI+ 1.1 records for processing or placement in the GDI+ Caches (see section [3.1.1.1.7](#Section_819a758e7ba24e6f96058b7b006cdad3)).

If the client has not advertised support for a particular cache type or feature, it SHOULD ignore any alternate secondary and primary drawing orders associated with that cache or feature, if they are sent by the server.

Processing of Create Offscreen Bitmap

The structure and fields of the Create Offscreen Bitmap Alternate Secondary Drawing Order are specified in section [2.2.2.2.1.3.2](#Section_427758ff4bac4833b388d634cc512d0c). The order fields MUST be processed in accordance with this description.

If the client does not support offscreen bitmaps (specified in the Offscreen Bitmap Cache Capability Set described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9), or the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders, this order SHOULD be ignored.

Processing of Switch Surface

The structure and fields of the Switch Surface Alternate Secondary Drawing Order are specified in section [2.2.2.2.1.3.3](#Section_ada4d8a9421e48c0a0ed7384fa3cf061). The order fields MUST be processed in accordance with this description.

The Create Offscreen Bitmap Alternate Secondary Drawing Order (see section [3.2.5.1.3.1.1](#Section_8594bbe257214cc4bf0ef03419acfa5d)), which is used to create the offscreen bitmap referenced in this order, MUST have been received by the client before processing this order.

If the client does not support offscreen bitmaps (specified in the Offscreen Bitmap Cache Capability Set described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9), or the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders, this order SHOULD be ignored.

The client SHOULD drop the connection if the value in the **bitmapId** field is less than SCREEN\_BITMAP\_SURFACE (0xFFFF) but greater than the maximum number of entries allowed in the Offscreen Bitmap Cache, as specified by the **offscreenCacheEntries** field of the Offscreen Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.9).

Processing of Create NineGrid Bitmap

The structure and fields of the Create NineGrid Bitmap Alternate Secondary Drawing Order are specified in section [2.2.2.2.1.3.4](#Section_6ac5eb820c30405490e883797f24bc38). The order fields MUST be processed in accordance with this description.

If the client does not support rendering NineGrid bitmaps (specified in the DrawNineGrid Capability Set described in section [2.2.1.2](#Section_c7fff28863db4521bbe577e060fb0780)), or support the DrawNineGrid and MultiDrawNineGrid Primary Drawing Orders (see sections [3.2.5.1.1.1.21](#Section_73caaa7677904cfa8bbe394106523a4a) and [3.2.5.1.1.1.22](#Section_c58568a997ae46a6945634411ccd1dee), respectively), this order SHOULD be ignored.

Processing of Stream Bitmap Orders

The structure and fields of the Stream Bitmap First and Stream Bitmap Next Alternate Secondary Drawing Orders are specified in sections [2.2.2.2.1.3.5.1](#Section_b714a7c97fd54621a397349a4ef3e06c) and [2.2.2.2.1.3.5.2](#Section_dfe099ffa894406393a7fbe64e7cb85e), respectively. The order fields MUST be processed in accordance with this description.

The Stream Bitmap Orders are only used to populate the NineGrid Bitmap Cache (see section [3.1.1.1.6](#Section_b71d528445c146e7b92511edd3b1de82)), and MUST follow immediately before the reception of a Create NineGrid Bitmap Alternate Secondary Drawing Order (see section [3.3.5.1.3.1.3](#Section_169f3f204c284f4caaf2ec071249d0e3)). The NineGrid Bitmap Cache entry to populate with the streamed bitmap data is implicitly assumed to be the entry specified in the Create NineGrid Bitmap Order.

If the client does not support rendering NineGrid bitmaps (specified in the DrawNineGrid Capability Set described in section [2.2.1.2](#Section_c7fff28863db4521bbe577e060fb0780)), or support the DrawNineGrid or MultiDrawNineGrid Primary Drawing Orders (see sections [3.2.5.1.1.1.21](#Section_73caaa7677904cfa8bbe394106523a4a) and [3.2.5.1.1.1.22](#Section_c58568a997ae46a6945634411ccd1dee), respectively), these orders SHOULD be ignored.

GDI+ Orders

Processing of Draw GDI+ Cache Orders

The structure and fields of the Draw GDI+ Cache First, Draw GDI+ Cache Next, and Draw GDI+ Cache End Alternate Secondary Drawing Orders are specified in sections [2.2.2.2.1.3.6.2](#Section_53dbc9c54e3f4b17a256baf43c582146), [2.2.2.2.1.3.6.3](#Section_20da0b5e8cc94805998e03e9f216bade), and [2.2.2.2.1.3.6.4](#Section_b44bd2b06c00454db182dbb841131c7a), respectively. The order fields MUST be processed in accordance with these descriptions.

If the client does not support rendering GDI+ 1.1 primitives (specified in the Draw GDI+ Capability Set Cache described in section [2.2.1.3](#Section_52635737d1444f479c88b48ceaf3efb4)), these orders SHOULD be ignored.

Processing of Draw GDI+ Orders

The structure and fields of the Draw GDI+ First, Draw GDI+ Next, and Draw GDI+ End Alternate Secondary Drawing Orders are specified in sections [2.2.2.2.1.3.6.5](#Section_03941c5e98d64f61ac74ddb307f1f317), [2.2.2.2.1.3.6.6](#Section_b5deca1d5802428998cb87035e1ca454), and [2.2.2.2.1.3.6.7](#Section_657e77cd01d94eccaf45a69e0b48390f), respectively. The order fields MUST be processed in accordance with these descriptions.

If the client does not support rendering GDI+ 1.1 primitives (specified in the Draw GDI+ Capability Set Cache described in section [2.2.1.3](#Section_52635737d1444f479c88b48ceaf3efb4)), these orders SHOULD be ignored.

#### Error Conditions

##### Sending of Bitmap Cache Error PDU

The Bitmap Cache Error PDU SHOULD be sent to a server that has requested the rendering of a cached bitmap that is not available in the client Bitmap Cache (see section [3.1.1.1.1](#Section_2bf9258842bd45278b3eb90c56e292d2)).

Rendering of items from the bitmap cache is accomplished using the [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) and [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Orders. If an error occurs during rendering, the client SHOULD complete the rendering operation by using a black bitmap, and then send a Bitmap Cache Error PDU to the server.

The structure and fields of the Bitmap Cache Error PDU are specified in section [2.2.2.3.1.1](#Section_484c7e89fc5f4f14b2d1f3d019d8a6b6), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.2.5.1 demonstrate how to initialize the contents of the PDU. The contents of this PDU MAY be compressed.

##### Sending of the Offscreen Bitmap Cache Error PDU

The Offscreen Bitmap Cache Error PDU SHOULD be sent to a server when the creation of an offscreen bitmap in the Offscreen Bitmap Cache (see section [3.1.1.1.5](#Section_63061daa980843c9b00dd5eda7dc8c21)) cannot be fulfilled due to a client-side failure (for example, low memory conditions).

Creation of an offscreen bitmap is accomplished by using the [Create Offscreen Bitmap (section 2.2.2.2.1.3.2)](#Section_427758ff4bac4833b388d634cc512d0c) Alternate Secondary Drawing Order. The client SHOULD send the Offscreen Bitmap Cache Error PDU to the server to request that it disable offscreen bitmap caching and resend the drawing updates associated with the affected area. Any further errors related to the offscreen bitmap caching MUST be ignored by the client.

The structure and fields of the Offscreen Bitmap Cache Error PDU are specified in section [2.2.2.3.2](#Section_c78f94f32a7f41a8902a1c6b30d34240), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.2.5.1 demonstrate how to initialize the contents of the PDU. The contents of this PDU MAY be compressed.

##### Sending of the DrawNineGrid Cache Error PDU

The DrawNineGrid Cache Error PDU SHOULD be sent to a server when the creation of a NineGrid bitmap in the NineGrid Bitmap Cache (see section [3.1.1.1.6](#Section_b71d528445c146e7b92511edd3b1de82)) cannot be fulfilled due to a client-side failure (for example, low memory conditions).

Creation of a NineGrid bitmap in the NineGrid Bitmap Cache is accomplished by first using the [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38) Alternate Secondary Drawing Order, and then streaming the bitmap into the cache using the [Stream Bitmap First (section 2.2.2.2.1.3.5.1)](#Section_b714a7c97fd54621a397349a4ef3e06c) and [Stream Bitmap Next (section 2.2.2.2.1.3.5.2)](#Section_dfe099ffa894406393a7fbe64e7cb85e) Alternate Secondary Drawing Orders. If processing of any of these orders fails, the client SHOULD send the DrawNineGrid Error PDU to the server to request that it disable NineGrid bitmap caching and resend the drawing updates associated with the affected area. Any further errors related to NineGrid bitmap caching MUST be ignored by the client.

The structure and fields of the DrawNineGrid Cache Error PDU are specified in section [2.2.2.3.3](#Section_e871fe47a339464696c555605067b260), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.2.5.1 demonstrate how to initialize the contents of the PDU. The contents of this PDU MAY be compressed.

##### Sending of the GDI+ Error PDU

The GDI+ Error PDU SHOULD be sent to a server when a GDI+ alternate secondary drawing order cannot be successfully processed due to a GDI+ rendering error or caching issue.

The client SHOULD send the GDI+ Error PDU to the server to request that it resend all GDI+ content as bitmaps and not rely on the local client side GDI+ rendering. (The six GDI+ PDUs that are used to cache and render GDI+ primitives are specified in section [2.2.2.2.1.3.6](#Section_095f82214abb4e6b9b38b50ceaa4a34e).) Any further errors related to GDI+ rendering SHOULD be ignored by the client.

The structure and fields of the GDI+ Error PDU are specified in section [2.2.2.3.4](#Section_e370124b1e9a4a079fc4db2a08a1e180), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.2.5.1 demonstrate how to initialize the contents of the PDU. The contents of this PDU MAY be compressed.

### Timer Events

None.

### Other Local Events

None.

## Server Details

### Abstract Data Model

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

**Note**  It is possible to implement the following conceptual data by using a variety of techniques as long as the implementation produces external behavior that is consistent with that described in this document.

#### Cached Bitmap Keys

The Cached Bitmap Keys store holds a collection of 64-bit bitmap keys, each of which uniquely identifies a bitmap image that has been sent to the client by using a Cache Bitmap (Revision 2) Secondary Drawing Order (section [2.2.2.2.1.2.3](#Section_3C76F8FBFCAA4CA69D2E3F449BDBED40)), or Cache Bitmap (Revision 3) Secondary Drawing Order (section [2.2.2.2.1.2.8](#Section_710CF5C26A0C4CEA8D62FA4F639D7C5A)).

#### Primary Drawing Order History

The Primary Drawing Order History store holds information on the fields that have been sent in primary drawing orders. There are three pieces of information that MUST be recorded:

* Last primary order type constructed
* Current bounding rectangle
* Per-order record of the last value used in each field

These records are updated as each primary drawing order is constructed, and are used to efficiently encode primary drawing orders for transmission on the wire using as few bytes as possible (see section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad)).

#### Bitmap Cache Wait List

The Bitmap Cache Wait List stores a collection of 64-bit identifiers, each of which identifies a bitmap that is a candidate for placement in the Bitmap Cache (see section [3.1.1.1.1](#Section_2bf9258842bd45278b3eb90c56e292d2)). The usage of the Bitmap Cache Wait List is specified using the Revision 2 Bitmap Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.4.2).

Bitmaps are placed into the Bitmap Cache Wait List the first time they are encountered in a server drawing operation. If any of the bitmaps in the wait list are encountered during a subsequent drawing operation, they MUST be removed from the wait list and placed into a Bitmap Cache.

Using a wait list in conjunction with persistent bitmap caching ensures that only bitmaps that are used more than once in rendering operations are written to persistent storage. In effect, it implies that a bitmap MUST be sent twice to the client before it is actually stored in a valid Bitmap Cache entry. (The first time it is sent to the client, it is used and not cached. The second time it is sent to the client, it is cached and then used.)

### Timers

None.

### Initialization

None.

### Higher-Layer Triggered Events

None.

### Processing Events and Sequencing Rules

#### Drawing Orders

All drawing orders are encapsulated in an [Orders Update (section 2.2.2.1)](#Section_ef7c27f5097746fea50ef9a51d6ab90b), which is sent as part of the Graphics Update PDU (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.1.3); or they are encapsulated in a [Fast-Path Orders Update (section 2.2.2.2)](#Section_7f53bb2892e94cbea15f8553c902e3c4), which is sent as part of the Fast-Path Update PDU (see [MS-RDPBCGR] section 2.2.9.1.2).

There are three classes of drawing orders:

* [Primary Drawing Orders (section 3.2.5.1.1)](#Section_90a5d432d45f4b4b96b3cdc3775e85f2)
* [Secondary Drawing Orders (section 2.2.2.2.1.2)](#Section_9df0ec924b494302aa863421151b0c11)
* [Alternate Secondary Drawing Orders (section 2.2.2.2.1.3)](#Section_b69a751e34df4326b4e10e582ff7ea97)

Orders belonging to each of these classes are packed together into an Orders Update structure or a Fast-Path Orders Update structure, each order being aligned on a byte boundary.

##### Primary Drawing Orders

###### Construction of a Primary Drawing Order

All primary drawing orders MUST conform to the structure and rules defined in section [2.2.2.2.1.1.2](#Section_23f766d483434e6b8281071ddccc0272).

To efficiently construct a primary drawing order, the server MUST use a [Primary Drawing Order History (section 3.2.1.1)](#Section_f1da06b585c940c7a0de3f280f480960) store. This store holds three pieces of information:

* Last primary order type constructed.
* Current bounding rectangle.
* Per-order record of the last value used in each field.

These stored records allow the server to use the minimum amount of data when constructing an order; if a field is unchanged from the value that it had when the order type was last sent, it SHOULD NOT be included in the order being constructed. Hence, only fields that have new values are required to be sent to the client. The fields that are present in the order MUST be indicated by the fieldFlags field.

If all of the Coord-type fields (see section [2.2.2.2.1.1.1.1](#Section_870b982c9abe40cb99d02c3f0cc5fb74)) in an order can be represented as a signed delta in the range -127 to 128 from the previous field value, the size of the order SHOULD be optimized by using delta-coordinates (see sections 2.2.2.2.1.1.1.1 and 2.2.2.2.1.1.2). In that case, all of the fields SHOULD be represented using delta-coordinates, and the TS\_DELTA\_COORDINATES (0x10) flag MUST be used in the primary drawing order header to indicate this fact.

Before a given order is sent, the server MUST also ensure that all of the data required to process the order is accessible to the client. For example, if the order refers to a cached item, that item MUST be present in the client-side cache when the order is processed. Or, if palettized color is being used, the correct palette MUST be applied at the client-side.

Once a primary drawing order has been constructed and transmitted to the client, the server MUST update the records in the [Primary Drawing Order History (section 3.3.1.2)](#Section_19b1b7f2456c4d928f5a5f507947889c) to ensure that future encodings use the minimum fields and data required.

Construction of DstBlt

The structure and fields of the DstBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.1](#Section_87ea30df59d6438ea73583f0225fbf91). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The DstBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of MultiDstBlt

The structure and fields of the MultiDstBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.2](#Section_aaae78403eb440389204b95f87b44534). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The MultiDstBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of PatBlt

The structure and fields of the PatBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.3](#Section_bd4bf5e7b98845f982013b22cc9aeeb8). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

If a cached brush is specified in this order, that brush MUST be sent to the client before this order is dispatched by using a Cache Brush Secondary Drawing Order (see section [3.3.5.1.2.1.6](#Section_739f70abcdd447c38033c77abb7080a5)). (The client specifies support for brush caching using the Brush Capability Set defined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7.)

The PatBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

Construction of MultiPatBlt

The structure and fields of the MultiPatBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.4](#Section_5f677576a62e48d9a58d711490882365). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

If a cached brush is specified in this order, that brush MUST be sent to the client before this order is dispatched by using a Cache Brush Secondary Drawing Order (see section [3.3.5.1.2.1.6](#Section_739f70abcdd447c38033c77abb7080a5)). (The client specifies support for brush caching using the Brush Capability Set defined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7.)

The MultiPatBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

Construction of OpaqueRect

The structure and fields of the OpaqueRect Primary Drawing Order are specified in section [2.2.2.2.1.1.2.5](#Section_1eead7aaac63411a9f8cb1b227526877). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The OpaqueRect Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of MultiOpaqueRect

The structure and fields of the MultiOpaqueRect Primary Drawing Order are specified in section [2.2.2.2.1.1.2.6](#Section_74e46cd9d6494bbe9f8af4200f23302e). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The MultiOpaqueRect Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of ScrBlt

The structure and fields of the ScrBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.7](#Section_a4e322b0cd644dfc8e1af24dc0edc99d). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The ScrBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of MultiScrBlt

The structure and fields of the MultiScrBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.8](#Section_31b79b4915b74709bbc2d58ce0362292). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The MultiScrBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of MemBlt

The structure and fields of the MemBlt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.9](#Section_84c2ec2ff776405b9b486894a28b1b14). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The source bitmap associated with the MemBlt Order MUST originate from one of three possible locations:

1. One of the bitmap caches (section [3.1.1.1.1](#Section_2bf9258842bd45278b3eb90c56e292d2))
2. The Offscreen Bitmap Cache (section [3.1.1.1.5](#Section_63061daa980843c9b00dd5eda7dc8c21))
3. The Bitmap Cache Wait List (section [3.3.1.3](#Section_fb7b28f140c94535bd60b4fd345d94c8))

If the source bitmap associated with the MemBlt Order refers to an item in one of the bitmap caches, the actual bitmap data MUST be sent to the client before this order is dispatched by using a Cache Bitmap Secondary Drawing Order (sections [3.3.5.1.2.1.1](#Section_30474912ea4d41338e7014d5c0ffd97e), [3.3.5.1.2.1.2](#Section_2575b398ca4848dea9aa887261a9e2c2), and [3.3.5.1.2.1.7](#Section_cb71c3b78acd48aca13fe7900e202ef3)).

If the source bitmap associated with the MemBlt Order refers to an item in the Offscreen Bitmap Cache, the actual entry MUST be created and initialized using the Create Offscreen Bitmap (section [3.3.5.1.3.1.1](#Section_fdd8d10d9f1e4897ad5f768d186707d1)) and Switch Surface (section [3.3.5.1.3.1.2](#Section_db7d6f301c574541b680d04807fbe1b4)) Alternate Secondary Drawing Orders; the Create Offscreen Bitmap Order creates the offscreen bitmap while the Switch Surface Order is used to redirect all drawing operations to the offscreen bitmap. The cacheIndex field of the MemBlt Order MUST be set to the index of the Offscreen Bitmap Cache entry, and the bitmap cache ID (specified as part of the cacheId field) MUST be set to TS\_BITMAPCACHE\_SCREEN\_ID (0xFF).

If the source bitmap associated with the MemBlt Order refers to a bitmap in the Bitmap Cache Wait List, the actual bitmap data MUST be sent to the client before this order is dispatched by using a Cache Bitmap (Revision 2) Secondary Drawing Order with the CBR2\_DO\_NOT\_CACHE (0x10) flag set (section 3.3.5.1.2.1.2), or a Cache Bitmap (Revision 3) Secondary Drawing Order with the CBR3\_DO\_NOT\_CACHE (0x10) flag set (section 3.3.5.1.2.1.7). The **cacheIndex** field of the MemBlt Order MUST be set to BITMAPCACHE\_WAITING\_LIST\_INDEX (32767), and the bitmap cache ID (specified as part of the **cacheId** field) MUST be set to the ID of the bitmap cache in which the bitmap will be stored when it is cached.

If palettized color is being used, the color table specified in the MemBlt Order MUST be sent to the client before this order is dispatched by using a Cache Color Table Secondary Drawing Order (section [3.3.5.1.2.1.3](#Section_a3900b17b26a4b66bca30969a5932dd9)). This step ensures that the client is able to render the cached bitmap correctly. Support for a Color Table Cache (section [3.1.1.1.3](#Section_b31800c6da774acc854bb954ebc1848b)) is implied by support for the MemBlt Order.

The MemBlt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of Mem3Blt

The structure and fields of the Mem3Blt Primary Drawing Order are specified in section [2.2.2.2.1.1.2.10](#Section_3aa21a4a031c4a3f8fc5159068f30237). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The construction of the Mem3Blt Order follows the same principles as those outlined for the MemBlt Order specified in section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac). However, the Mem3Blt Order includes fields to specify a brush.

If a cached brush is specified in this order, that brush MUST be sent to the client before this order is dispatched by using a Cache Brush Secondary Drawing Order (see section [3.3.5.1.2.1.6](#Section_739f70abcdd447c38033c77abb7080a5)). (The client specifies support for brush caching using the Brush Capability Set defined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7.)

The Mem3Blt Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

Construction of LineTo

The structure and fields of the LineTo Primary Drawing Order are specified in section [2.2.2.2.1.1.2.11](#Section_27276fcf32be49a99a1797ec6167ac7e). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The LineTo Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of SaveBitmap

The structure and fields of the SaveBitmap Primary Drawing Order are specified in section [2.2.2.2.1.1.2.12](#Section_db4ce8dbcf76483d99ba354cd010ea4a). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The SaveBitmap Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of GlyphIndex

The structure and fields of the GlyphIndex Primary Drawing Order are specified in section [2.2.2.2.1.2.5](#Section_6dc06088b1124f4fb0bcb97bba621a0c). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

All of the glyphs associated with the glyph cache indices specified in the order MUST be sent to the client before this order is dispatched by using a Revision 1 or 2 Cache Glyph Secondary Drawing Order (see sections [3.3.5.1.2.1.1](#Section_30474912ea4d41338e7014d5c0ffd97e) and [3.3.5.1.2.1.2](#Section_2575b398ca4848dea9aa887261a9e2c2)) or a FastGlyph Primary Drawing Order (see section [3.3.5.1.1.1.15](#Section_a2f3f164455e4422b742a5f9e6e4a0ab)). The usage of glyph cache indices implies that support for glyph caching MUST have been specified in the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8). If a fragment cache index is specified in a USE clause, that fragment MUST have been part of an ADD clause in a prior GlyphIndex or FastIndex order.

If a cached brush is specified in this order, that brush MUST be sent to the client before this order is dispatched by using a Cache Brush Secondary Drawing Order (see section [3.3.5.1.2.1.6](#Section_739f70abcdd447c38033c77abb7080a5)). (The client specifies support for brush caching using the Brush Capability Set defined in [MS-RDPBCGR] section 2.2.7.1.7.)

If support for the FastIndex Primary Drawing Order (see section [2.2.2.2.1.1.2.14](#Section_ac55f919acb943c1a3ec33802929f9f7) ) was indicated in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3), then the FastIndex Primary Drawing Order SHOULD be used to send the glyph indices.

Construction of FastIndex

The structure and fields of the FastIndex Primary Drawing Order are specified in section [2.2.2.2.1.1.2.14](#Section_ac55f919acb943c1a3ec33802929f9f7). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The construction of the FastIndex Order follows the same principles as those outlined for the GlyphIndex Order specified in section [3.3.5.1.1.1.13](#Section_c6e78626f6eb4f17897f2da11847ff89). However, the FastIndex Order does not use cached brushes and also utilizes a more efficient field encoding.

The FastIndex Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of FastGlyph

The structure and fields of the FastGlyph Primary Drawing Order are specified in section [2.2.2.2.1.1.2.15](#Section_839e963ce24a42adb4bdeb108340629c). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

All of the glyphs associated with the glyph cache indices specified in the order MUST be sent to the client before this order is dispatched by using a Revision 1 or 2 Cache Glyph Secondary Drawing Order (see sections [2.2.2.2.1.2.2](#Section_c9365b320be547c9af1afeba2ea1ee9f) and [2.2.2.2.1.2.3](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40)) or a prior FastGlyph Primary Drawing Order. The usage of glyph cache indices implies that support for glyph caching MUST have been specified in the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8).

The FastGlyph Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

Construction of PolygonSC

The structure and fields of the PolygonSC Primary Drawing Order are specified in section [2.2.2.2.1.1.2.16](#Section_22e1117fd40d41df970959389ac1f8dc). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The PolygonSC Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of PolygonCB

The structure and fields of the PolygonCB Primary Drawing Order are specified in section [2.2.2.2.1.1.2.17](#Section_24f4e2291f9f4633b9be66d7579ba36d). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

If a cached brush is specified in this order, that brush MUST be sent to the client before this order is dispatched by using a Cache Brush Secondary Drawing Order (see section [3.3.5.1.2.1.6](#Section_739f70abcdd447c38033c77abb7080a5)). (The client specifies support for brush caching using the Brush Capability Set defined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7.)

The PolygonCB Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

Construction of PolyLine

The structure and fields of the Polyline Primary Drawing Order are specified in section [2.2.2.2.1.1.2.18](#Section_b60b4d9baac54b9884d3edc0c0779a5c). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The Polyline Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of EllipseSC

The structure and fields of the EllipseSC Primary Drawing Order are specified in section [2.2.2.2.1.1.2.19](#Section_5146fab3c43f44d7b54104666a80e33b). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The EllipseSC Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of EllipseCB

The structure and fields of the EllipseCB Primary Drawing Order are specified in section [2.2.2.2.1.1.2.20](#Section_897203838394423a810aba631d66055f). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

If a cached brush is specified in this order, that brush MUST be sent to the client before this order is dispatched by using a Cache Brush Secondary Drawing Order (see section [3.3.5.1.2.1.6](#Section_739f70abcdd447c38033c77abb7080a5)). (The client specifies support for brush caching using the Brush Capability Set defined in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7.)

The EllipseCB Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [MS-RDPBCGR] section 2.2.7.1.3).

Construction of DrawNineGrid

The structure and fields of the DrawNineGrid Primary Drawing Order are specified in section [2.2.2.2.1.1.2.21](#Section_44bd6a68a5b74a38980706ba4ab4a72a). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

Support for the NineGrid Bitmap Cache (see section [3.1.1.1.6](#Section_b71d528445c146e7b92511edd3b1de82)) MUST have been specified using the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780) because the order refers to the ID of a NineGrid-compatible bitmap in this cache. Furthermore, this bitmap MUST have been created and initialized before this order is dispatched by using the Create NineGrid Bitmap (see section [3.3.5.1.3.1.3](#Section_169f3f204c284f4caaf2ec071249d0e3)) and Stream Bitmap (see section [3.3.5.1.3.1.4](#Section_9f8b5bdc78c146cbb84cbe03516ae7a6)) Alternate Secondary Orders.

The DrawNineGrid Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of MultiDrawNineGrid

The structure and fields of the MultiDrawNineGrid Primary Drawing Order are specified in section [2.2.2.2.1.1.2.22](#Section_101fecdc3ad841c6b7b670d8efd4ad70). The order fields MUST be populated in accordance with this description and the instructions detailed in section [3.3.5.1.1.1](#Section_4c27bb5935d74c4fbc91908d6bc6cdad).

The construction of the MultiDrawNineGrid Order follows the same principles as those outlined for the DrawNineGrid order specified in section [3.3.5.1.1.1.21](#Section_ddb66f29ea824d54ad94992e3701f9a5). However, the MultiDrawNineGrid includes multiple clipping rectangles, as opposed to the DrawNineGrid Order, which only includes one clipping rectangle.

The MultiDrawNineGrid Order MUST NOT be sent to the client if support for it was not specified in the Order Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

##### Secondary Drawing Orders

###### Construction of Secondary Drawing Orders

All secondary drawing orders are identified by the [Secondary Drawing Order Header (section 2.2.2.2.1.2.1.1)](#Section_c54e2667335d4a59a6e27dc9744dbe79). Client support for a given secondary order is determined by the capabilities advertised by the client.

Construction of Cache Bitmap (Revision 1)

The structure and fields of the Cache Bitmap (Revision 1) Secondary Drawing Order are specified in section [2.2.2.2.1.2.2](#Section_c9365b320be547c9af1afeba2ea1ee9f). The order fields MUST be populated in accordance with this description.

The bitmaps that are cached by the Cache Bitmap (Revision 1) Secondary Drawing Order can be consumed by future MemBlt (see section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac)) and Mem3Blt (see section [3.3.5.1.1.1.10](#Section_8079e790fe444aa5a95a6b0d43860517)) Primary Drawing Orders.

The bitmap data MUST NOT include a Compressed Data Header structure (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.1.3.1.2.3) if the exclusion of this header was specified in the General Capability Set (see [MS-RDPBCGR] section 2.2.7.1.1).

The cache ID and cache index to use are selected by the server based on the management policies that are implemented (for example, an LRU policy might be in effect).

The Cache Bitmap (Revision 1) Order MUST NOT be sent to the client if support for bitmap caching was not specified using the Revision 1 Bitmap Cache Capability Set (see [MS-RDPBCGR] section 2.2.7.1.4.1). Furthermore, if client-side support for the MemBlt or Mem3Blt Primary Drawing Orders (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3) does not exist, the Cache Bitmap (Revision 1) Order SHOULD NOT be sent to the client.

Construction of Cache Bitmap (Revision 2)

The structure and fields of the Cache Bitmap (Revision 2) Secondary Drawing Order are specified in section [2.2.2.2.1.2.3](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40). The order fields MUST be populated in accordance with this description.

The bitmaps that are cached by the Cache Bitmap (Revision 2) Secondary Drawing Order can be consumed by future MemBlt (see section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac)) and Mem3Blt (see section [3.3.5.1.1.1.11](#Section_eab5d815037848258da3b5d2cb718b2f)) Primary Drawing Orders.

The bitmap data MUST NOT include a Compressed Data Header structure (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.9.1.1.3.1.2.3) if the exclusion of this header was specified in the General Capability Set (see [MS-RDPBCGR] section 2.2.7.1.1).

The cache ID and cache index to use are selected by the server based on the management policies that are implemented (for example, an LRU policy might be in effect).

If a 64-bit key identifying the bitmap MUST be generated for the bitmap (dictated by the capabilities for the bitmap cache specified in the Revision 2 Bitmap Cache Capability Set as defined in [MS-RDPBCGR] section 2.2.7.1.4.2), the server MUST create a unique 64-bit identifier and include it in the **Key1** and **Key2** fields of the Cache Bitmap (Revision 2) Order.

If it is the first time the server is instructing the client to cache the bitmap, and a Bitmap Cache Wait List (see section [3.3.1.3](#Section_fb7b28f140c94535bd60b4fd345d94c8)) is present, the server MUST force the bitmap into the Wait List and set the CBR2\_DO\_NOT\_CACHE (0x10) flag in the Cache Bitmap (Revision 2) Secondary Drawing Order. The **cacheId** field MUST be set to the ID of the bitmap cache in which the bitmap will be stored if it is encountered again, and the **cacheIndex** field MUST be set to BITMAPCACHE\_WAITING\_LIST (32767). If the bitmap is encountered again, it MUST be removed from the Wait List and sent to the client without the CBR2\_DO\_NOT\_CACHE flag.

The Cache Bitmap (Revision 2) Order MUST NOT be sent to the client if support for bitmap caching was not specified using the Revision 2 Bitmap Cache Capability Set (see [MS-RDPBCGR] section 2.2.7.1.4.2). Furthermore, if client-side support for the MemBlt (see section 3.3.5.1.1.1.9) and Mem3Blt (see section [3.3.5.1.1.1.10](#Section_8079e790fe444aa5a95a6b0d43860517)) Primary Drawing Orders (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3) does not exist, the Cache Bitmap (Revision 2) Order SHOULD NOT be sent to the client.

Construction of Cache Color Table

The structure and fields of the Cache Color Table Secondary Drawing Order are specified in section [2.2.2.2.1.2.4](#Section_49fad30181714f38a68a7394baeb7b15). The order fields MUST be populated in accordance with this description.

The color tables that are cached by the Cache Color Table Secondary Drawing Order can be consumed by future MemBlt (see section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac)) and Mem3Blt (see section [3.3.5.1.1.1.11](#Section_eab5d815037848258da3b5d2cb718b2f)) Primary Drawing Orders.

The cache ID and cache index to use are selected by the server based on the management policies that are implemented (for example, an LRU policy might be in effect).

The Cache Color Table Order SHOULD NOT be sent to the client if it does not support the MemBlt and Mem3Blt Primary Drawing Orders (specified using the Order Capability Set specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3).

Construction of Cache Glyph (Revision 1)

The structure and fields of the Cache Glyph (Revision 1) Secondary Drawing Order are specified in section [2.2.2.2.1.2.5](#Section_6dc06088b1124f4fb0bcb97bba621a0c). The order fields MUST be populated in accordance with this description.

The glyphs that are cached by the Cache Glyph (Revision 1) Secondary Drawing Order can be consumed by future GlyphIndex (see section [3.3.5.1.1.1.13](#Section_c6e78626f6eb4f17897f2da11847ff89)), FastIndex (see section [3.3.5.1.1.1.14](#Section_a95fccd6a18e460ba2bc7c8776dfced7)), and FastGlyph (see section [3.3.5.1.1.1.15](#Section_a2f3f164455e4422b742a5f9e6e4a0ab)) Primary Drawing Orders.

The cache index to use is selected by the server based on the management policies that are implemented (for example, an LRU policy might be in effect).

The Cache Glyph (Revision 1) Order MUST NOT be sent to the client if support for Revision 1 glyph caching was not specified using the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8). Furthermore, if client-side support for the GlyphIndex, FastIndex, or FastGlyph Primary Drawing Orders (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3) does not exist, the Cache Glyph (Revision 1) Order SHOULD NOT be sent to the client.

Construction of Cache Glyph (Revision 2)

The structure and fields of the Cache Glyph (Revision 2) Secondary Drawing Order are specified in section [2.2.2.2.1.2.6](#Section_ffc80435136e442e908540ca1d27a413). The order fields MUST be populated in accordance with this description.

The construction of the Cache Glyph (Revision 2) Order follows the same principles as those outlined for the Cache Glyph (Revision 1) Order specified in section [3.3.5.1.2.1.4](#Section_d732695349e1406298e0ea620c7ee00f). However, the Cache Glyph (Revision 2) Order utilizes a more efficient field encoding.

The Cache Glyph (Revision 2) Order MUST NOT be sent to the client if support for Revision 2 glyph caching was not specified using the Glyph Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.8). Furthermore, if client-side support for the GlyphIndex, FastIndex, and FastGlyph Primary Drawing Orders (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3) does not exist, the Cache Glyph (Revision 2) Order SHOULD NOT be sent to the client.

Construction of Cache Brush

The structure and fields of the Cache Brush Secondary Drawing Order are specified in section [2.2.2.2.1.2.7](#Section_9e7027c4d9894cb09674e2e36c95ebdd). The order fields MUST be populated in accordance with this description.

The brushes that are cached by the Cache Brush Secondary Drawing Order can be consumed by a number of primary drawing orders:

* PatBlt (see section [3.3.5.1.1.1.3](#Section_c0574e4a14354b9da4ad627d02537640))
* MultiPatBlt (see section [3.3.5.1.1.1.4](#Section_36d970a8075a4fa2ac73d4dd579c4f83))
* Mem3Blt (see section [3.3.5.1.1.1.10](#Section_8079e790fe444aa5a95a6b0d43860517))
* GlyphIndex (see section [3.3.5.1.1.1.13](#Section_c6e78626f6eb4f17897f2da11847ff89))
* PolygonCB (see section [3.3.5.1.1.1.17](#Section_4709860f9fef4d268cda26dc2bd2576e))
* EllipseCB (see section [3.3.5.1.1.1.20](#Section_88be0cea5ba141419acd74fc3473577f))

The cache index to use is selected by the server based on the management policies that are implemented (for example, an LRU policy might be in effect).

The Cache Brush Order MUST NOT be sent to the client if support for brush caching was not specified using the Brush Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.7). Furthermore, if the client does not support any primary drawing orders that use cached brushes, the Cache Brush Order SHOULD NOT be sent to the client.

Construction of Cache Bitmap (Revision 3)

The structure and fields of the Cache Bitmap (Revision 3) Secondary Drawing Order are specified in section [2.2.2.2.1.2.8](#Section_710cf5c26a0c4cea8d62fa4f639d7c5a). The order fields MUST be populated in accordance with this description.

The bitmaps that are cached by the Cache Bitmap (Revision 3) Secondary Drawing Order can be consumed by future MemBlt (section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac)) and Mem3Blt (section [3.3.5.1.1.1.10](#Section_8079e790fe444aa5a95a6b0d43860517)) Primary Drawing Orders.

The cache ID and cache index to use are selected by the server based on the management policies that are implemented (for example, an LRU policy might be in effect). The server MUST create a unique 64-bit identifier and include it in the **Key1** and **Key2** fields of the Cache Bitmap (Revision 3) Order.

If it is the first time the server is instructing the client to cache the bitmap, and a Bitmap Cache Wait List (see section [3.3.1.3](#Section_fb7b28f140c94535bd60b4fd345d94c8)) is present, the server MUST force the bitmap into the Wait List and set the CBR3\_DO\_NOT\_CACHE (0x10) flag in the Cache Bitmap (Revision 3) Secondary Drawing Order. The **cacheId** field MUST be set to the ID of the bitmap cache in which the bitmap will be stored if it is encountered again, and the **cacheIndex** field MUST be set to BITMAPCACHE\_WAITING\_LIST (32767). If the bitmap is encountered again, it MUST be removed from the Wait List and sent to the client without the CBR3\_DO\_NOT\_CACHE flag.

The client advertises support for the Cache Bitmap (Revision 3) Order by including the ORDERFLAGS\_EX\_CACHE\_BITMAP\_REV3\_SUPPORT (0x0002) flag in the **orderSupportExFlags** field of the Order Capability Set ([[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3). If a client does not include this flag, then the Cache Bitmap (Revision 3) Order MUST NOT be sent to the client.

The Cache Bitmap (Revision 3) Order MUST NOT be sent to the client if support for bitmap caching was not specified using the Revision 2 Bitmap Cache Capability Set ([MS-RDPBCGR] section 2.2.7.1.4.2). Furthermore, if client-side support for the MemBlt (section 3.3.5.1.1.1.9) and Mem3Blt (section 3.3.5.1.1.1.10) Primary Drawing Orders (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3) does not exist, the Cache Bitmap (Revision 2) Order SHOULD NOT be sent to the client.

##### Alternate Secondary Drawing Orders

###### Construction of Alternate Secondary Drawing Orders

All alternate secondary drawing orders MUST contain the [Alternate Secondary Drawing Order Header (section 2.2.2.2.1.3.1.1)](#Section_b93614406e084a0697e76e1b7a4cc204). Client support for a given alternate secondary order is determined by the capabilities advertised by the client.

Construction of Create Offscreen Bitmap

The structure and fields of the Create Offscreen Bitmap Alternate Secondary Drawing Order are specified in section [2.2.2.2.1.3.2](#Section_427758ff4bac4833b388d634cc512d0c). The order fields MUST be populated in accordance with this description.

The offscreen bitmaps managed by the Create Offscreen Bitmap Alternate Secondary Drawing Order are specified in the Switch Surface Alternate Secondary Drawing Order (see section [3.3.5.1.3.1.2](#Section_db7d6f301c574541b680d04807fbe1b4)). The Switch Surface Order allows the server to change the default client rendering surface to any one of the bitmaps created in the Offscreen Bitmap Cache by the Create Offscreen Order. Once drawing to an offscreen bitmap is complete, the server MUST direct the client to render the bitmap data to the primary drawing surface by using the MemBlt (see section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac)) or Mem3Blt (see section [3.3.5.1.1.1.10](#Section_8079e790fe444aa5a95a6b0d43860517)) Primary Drawing Order.

The Create Offscreen Bitmap Order MUST NOT be sent to the client if support for offscreen bitmap caching was not specified using the Offscreen Bitmap Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9), or if offscreen bitmap caching has been disabled due to the reception of an Offscreen Bitmap Cache Error PDU (see section [3.3.5.2.2](#Section_251bb5f061064d9091db82047661561e)). Furthermore, if client-side support for the MemBlt and Mem3Blt Primary Drawing Orders does not exist (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3), the Create Offscreen Bitmap Order SHOULD NOT be sent to the client.

Construction of Switch Surface

The structure and fields of the Switch Surface Alternate Secondary Drawing Order are specified in section [2.2.2.2.1.3.3](#Section_ada4d8a9421e48c0a0ed7384fa3cf061). The order fields MUST be populated in accordance with this description.

The Create Offscreen Bitmap Alternate Secondary Drawing Order (see section [3.3.5.1.3.1.1](#Section_fdd8d10d9f1e4897ad5f768d186707d1)), which is used to create the offscreen bitmap referenced in this order, MUST be sent to the client before this order is dispatched.

The Switch Surface Order MUST NOT be sent to the client if support for offscreen bitmap caching was not specified using the Offscreen Bitmap Cache Capability Set (see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.9), or if offscreen bitmap caching has been disabled due to the reception of an Offscreen Bitmap Cache Error PDU (see section [3.3.5.2.2](#Section_251bb5f061064d9091db82047661561e)). Furthermore, if client-side support for the MemBlt (see section [3.3.5.1.1.1.9](#Section_e2bed8c4ba844801a28d1d42353323ac)) or Mem3Blt (see section [3.3.5.1.1.1.10](#Section_8079e790fe444aa5a95a6b0d43860517)) Primary Drawing Order does not exist (specified using the Order Capability Set specified in [MS-RDPBCGR] section 2.2.7.1.3), the Switch Surface Order SHOULD NOT be sent to the client.

Construction of Create NineGrid Bitmap

The structure and fields of the Create NineGrid Alternate Secondary Drawing Order are specified in section [2.2.2.2.1.3.4](#Section_6ac5eb820c30405490e883797f24bc38). The order fields MUST be populated in accordance with this description.

The NineGrid bitmaps produced by the Create NineGrid Bitmap Alternate Secondary Drawing Order are initialized by the Stream Bitmap Alternate Secondary Drawing Orders (see section [3.3.5.1.3.1.4](#Section_9f8b5bdc78c146cbb84cbe03516ae7a6)) and consumed by the DrawNineGrid and MultiDrawNineGrid Primary Drawing Orders (see sections [3.3.5.1.1.1.21](#Section_ddb66f29ea824d54ad94992e3701f9a5) and [3.3.5.1.1.1.22](#Section_b65939940f4f4550885222b97b7db363)).

The Create NineGrid Bitmap Order MUST NOT be sent to the client if support for NineGrid rendering was not specified using the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780), or if NineGrid bitmap caching has been disabled due to the reception of a DrawNineGrid Cache Error PDU (see section [3.3.5.2.3](#Section_1333ae9693494b8586ab6726b652a06c)). Furthermore, if client-side support for the DrawNineGrid and MultiDrawNineGrid Primary Drawing Orders does not exist (specified using the Order Capability Set specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the Create NineGrid Bitmap Order SHOULD NOT be sent to the client.

Construction of Stream Bitmap Orders

The structure and fields of the Stream Bitmap First and Stream Bitmap Next Alternate Secondary Drawing Orders are specified in sections [2.2.2.2.1.3.5.1](#Section_b714a7c97fd54621a397349a4ef3e06c) and [2.2.2.2.1.3.5.2](#Section_dfe099ffa894406393a7fbe64e7cb85e), respectively. The order fields MUST be populated in accordance with these descriptions.

The Stream Bitmap Orders are only used to populate the NineGrid Bitmap Cache (see section [3.1.1.1.6](#Section_b71d528445c146e7b92511edd3b1de82)) and MUST follow immediately after the Create NineGrid Bitmap Alternate Secondary Drawing Order (see section [3.3.5.1.3.1.3](#Section_169f3f204c284f4caaf2ec071249d0e3)). The NineGrid Bitmap Cache entry to populate with the streamed bitmap data is implicitly assumed to be the entry specified in the Create NineGrid Bitmap Order.

Because the Stream Bitmap Orders are only used to populate the NineGrid Bitmap Cache, they SHOULD NOT be sent to the client if support for NineGrid rendering was not specified using the [DrawNineGrid Cache Capability Set (section 2.2.1.2)](#Section_c7fff28863db4521bbe577e060fb0780), or if NineGrid bitmap caching has been disabled due to the reception of a DrawNineGrid Cache Error PDU (see section [3.3.5.2.3](#Section_1333ae9693494b8586ab6726b652a06c)). Furthermore, if client-side support for the DrawNineGrid and MultiDrawNineGrid Primary Drawing Orders does not exist (specified using the Order Capability Set specified in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 2.2.7.1.3), the Stream Bitmap Orders SHOULD NOT be sent to the client.

GDI+ Orders

Construction of Draw GDI+ Cache Orders

The structure and fields of the Draw GDI+ Cache First, Draw GDI+ Cache Next, and Draw GDI+ Cache End Alternate Secondary Drawing Orders are specified in sections [2.2.2.2.1.3.6.2](#Section_53dbc9c54e3f4b17a256baf43c582146), [2.2.2.2.1.3.6.3](#Section_20da0b5e8cc94805998e03e9f216bade), and [2.2.2.2.1.3.6.4](#Section_b44bd2b06c00454db182dbb841131c7a), respectively. The order fields MUST be populated in accordance with these descriptions.

GDI+ 1.1 primitives cached by the Draw GDI+ Cache Orders are consumed by the Draw GDI+ Orders (see section [3.3.5.1.3.1.5.2](#Section_1f5a15d985664a81a6ff3a729b28b454)).

The Draw GDI+ Cache Orders MUST NOT be sent to the client if support for GDI+ 1.1 rendering was not specified using the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4), or if GDI+ 1.1 rendering has been disabled due to the reception of a GDI+ Error PDU (see section [3.3.5.2.4](#Section_cb29f62352484d2bbcede163df640729)).

Construction of Draw GDI+ Orders

The structure and fields of the Draw GDI+ First, Draw GDI+ Next, and Draw GDI+ End Alternate Secondary Drawing Orders are specified in sections [2.2.2.2.1.3.6.5](#Section_03941c5e98d64f61ac74ddb307f1f317), [2.2.2.2.1.3.6.6](#Section_b5deca1d5802428998cb87035e1ca454), and [2.2.2.2.1.3.6.7](#Section_657e77cd01d94eccaf45a69e0b48390f), respectively. The order fields MUST be populated in accordance with these descriptions.

The Draw GDI+ Orders MUST NOT be sent to the client if support for GDI+ 1.1 rendering was not specified using the [Draw GDI+ Capability Set (section 2.2.1.3)](#Section_52635737d1444f479c88b48ceaf3efb4), or if GDI+ 1.1 rendering has been disabled due to the reception of a GDI+ Error PDU (see section [3.3.5.2.4](#Section_cb29f62352484d2bbcede163df640729)).

#### Error Conditions

##### Processing of Bitmap Cache Error PDU

The structure and fields of the Bitmap Cache Error PDU are specified in section [2.2.2.3.1](#Section_fdc28a58010549caa9040fe3bee58ec2), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.3.5.2 demonstrate how to process the contents of the PDU.

Once this PDU has been processed, the server MUST flush the appropriate Bitmap Cache entries (see section [3.1.1.1.1](#Section_2bf9258842bd45278b3eb90c56e292d2)) and resend the graphics data associated with the affected area.

The server SHOULD honor up to five Bitmap Cache Error PDUs for a given connection; further error PDUs SHOULD be ignored for the duration of the connection to reduce the server overhead.

##### Processing of the Offscreen Bitmap Cache Error PDU

The structure and fields of the Offscreen Bitmap Cache Error PDU are specified in section [2.2.2.3.2](#Section_c78f94f32a7f41a8902a1c6b30d34240), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.3.5.2 demonstrate how to process the contents of the PDU.

Once this PDU has been processed, the server MUST disable offscreen bitmap caching for the duration of the connection and resend the graphics data associated with the affected area.

Once offscreen bitmap caching has been disabled, the server MUST NOT send the Create Offscreen Bitmap Alternate Secondary Drawing Order (see section [2.2.2.2.1.3.2](#Section_427758ff4bac4833b388d634cc512d0c)) for the duration of the connection.

##### Processing of the DrawNineGrid Cache Error PDU

The structure and fields of the DrawNineGrid Cache Error PDU are specified in section [2.2.2.3.3](#Section_e871fe47a339464696c555605067b260), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.3.5.2 demonstrate how to process the contents of the PDU.

Once this PDU has been processed, the server MUST disable NineGrid bitmap caching for the duration of the connection and resend the graphics data associated with the affected area.

Once NineGrid bitmap caching has been disabled, the server MUST NOT send the [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38) Alternate Secondary Drawing Order, and the [Stream Bitmap First (section 2.2.2.2.1.3.5.1)](#Section_b714a7c97fd54621a397349a4ef3e06c) and [Stream Bitmap Next (section 2.2.2.2.1.3.5.2)](#Section_dfe099ffa894406393a7fbe64e7cb85e) Alternate Secondary Drawing Orders for the duration of the connection.

##### Processing of the GDI+ Error PDU

The structure and fields of the GDI+ Error PDU are specified in section [2.2.2.3.4](#Section_e370124b1e9a4a079fc4db2a08a1e180), and the techniques described in [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 3.3.5.2 demonstrate how to process the contents of the PDU.

Once this PDU has been processed, the server MUST disable GDI+ 1.1 rendering for the duration of the connection. All future GDI+ content MUST be sent as bitmaps so that local client-side GDI+ rendering is not required.

Once GDI+ 1.1 rendering has been disabled, the six GDI+ PDUs that are used to cache and render GDI+ primitives (described in section [2.2.2.2.1.3.6](#Section_095f82214abb4e6b9b38b50ceaa4a34e)) MUST NOT be sent by the server for the duration of the connection.

### Timer Events

None.

### Other Local Events

None.

# Protocol Examples

## Annotated Primary Drawing Orders

### DstBlt

The following is an annotated dump of a [DstBlt (section 2.2.2.2.1.1.2.1)](#Section_87ea30df59d6438ea73583f0225fbf91) Primary Drawing Order.

1. 00000000 09 00 0c 48 00 37 01 ...H.7.
2. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
3. 0x09
4. = 0x01 |
5. 0x08
6. = TS\_STANDARD |
7. TS\_TYPE\_CHANGE
8. 00 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_DSTBLT\_ORDER
9. 0c -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x0c
10. Binary of 0x0c = 0000 1100
11. Fields 3, 4 are present
12. DSTBLT\_ORDER::nLeftRect not present
13. DSTBLT\_ORDER::nTopRect not present
14. 48 00 -> DSTBLT\_ORDER::nWidth = 0x48 = 72
15. 37 01 -> DSTBLT\_ORDER::nHeight = 0x0137 = 311
16. DSTBLT\_ORDER::bRop not present

### MultiDstBlt

The following is an annotated dump of a [MultiDstBlt (section 2.2.2.2.1.1.2.2)](#Section_aaae78403eb440389204b95f87b44534) Primary Drawing Order.

1. 00000000 09 0f 7f 12 01 2d 01 a0 00 0c 00 55 02 0d 00 04 .....-.....U....
2. 00000010 81 12 81 2d 80 58 0c 80 60 80 40 0c ...-.X..`.@.
3. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
4. 0x09
5. = 0x01 |
6. 0x08
7. = TS\_STANDARD |
8. TS\_TYPE\_CHANGE
9. 0f -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_MULTIDSTBLT\_ORDER
10. 7f -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x7f
11. Binary of 0x7f = 0111 1111
12. Fields 1-7 are present
13. 12 01 -> MULTI\_DSTBLT\_ORDER:nLeftRect = 0x0112 = 274
14. 2d 01 -> MULTI\_DSTBLT\_ORDER:nTopRect = 0x012d = 301
15. a0 00 -> MULTI\_DSTBLT\_ORDER::nWidth = 0xa0 = 160
16. 0c 00 -> MULTI\_DSTBLT\_ORDER::nHeight = 0x0c = 12
17. 55 -> MULTI\_DSTBLT\_ORDER::bRop = 0x55 = ROP Table Entry #85 = 0x00550009
18. 02 -> MULTI\_DSTBLT\_ORDER::nDeltaEntries = 0x2 = 2 entries
19. 0d 00 -> VARIABLE2\_FIELD::cbData = 0xd = 13 bytes
20. 04 81 12 81 2d 80 58 0c 80 60 80 40 0c -> VARIABLE2\_FIELD::rgbData
21. 04 -> DELTA\_RECTS\_FIELD::zeroBits = binary:0000 0100
22. Rectangle #1:
23. 81 12 -> Delta Left = 0x112 = 274
24. 81 2d -> Delta Top = 0x12d = 301
25. 80 58 -> Delta Width = 0x58 = 88
26. 0c -> Delta Height = 0xc = 12
27. Rectangle is (274, 301, 274 + 88 = 362, 301 + 12 = 313)
28. Rectangle #2:
29. 80 60 -> Delta Left = 0x60 = 96
30. Delta Top = 0 (zeroBit is 1)
31. 80 40 -> Delta Width = 0x40 = 64
32. 0c -> Delta Height = 12
33. Rectangle is (274 + 96 = 370, 301 + 0 = 301, 370 + 64 = 434, 301 + 12 = 313)

### PatBlt

The following is an annotated dump of a [PatBlt (section 2.2.2.2.1.1.2.3)](#Section_bd4bf5e7b98845f982013b22cc9aeeb8) Primary Drawing Order.

1. 00000000 09 01 7f 02 1a 00 c3 01 0d 00 0d 00 f0 ff ff 00 ................
2. 00000010 5b ef 00 81 [...
3. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
4. 0x09
5. = 0x01 |
6. 0x08
7. = TS\_STANDARD |
8. TS\_TYPE\_CHANGE
9. 01 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_PATBLT\_ORDER
10. 7f 02 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x027f
11. Binary of 0x7f = 0000 0010 0111 1111
12. Fields 1-7, 10 are present
13. 1a 00 -> PATBLT\_ORDER::nLeftRect = 0x1a = 26
14. c3 01 -> PATBLT\_ORDER::nTopRect = 0x01c3 = 451
15. 0d 00 -> PATBLT\_ORDER::nWidth = 0x0d = 13
16. 0d 00 -> PATBLT\_ORDER::nHeight = 0x0d = 13
17. f0 -> PATBLT\_ORDER::bRop = 0xf0 = ROP Table Entry #240 = 0x00f00021
18. ff ff 00 -> PATBLT\_ORDER::BackColor
19. TS\_COLOR::RedOrPaletteIndex = 0xff
20. TS\_COLOR::Green = 0xff
21. TS\_COLOR::Blue = 0x00
22. 5b ef 00 -> PATBLT\_ORDER::ForeColor
23. TS\_COLOR::RedOrPaletteIndex = 0x5b
24. TS\_COLOR::Green = 0xef
25. TS\_COLOR::Blue = 0x00
26. PATBLT\_ORDER::BrushOrgX not present
27. PATBLT\_ORDER::BrushOrgY not present
28. 81 --> PATBLT\_ORDER::BrushStyle = 0x81
29. 0x81
30. = 0x01 |
31. 0x80
32. = BMF\_1BPP |
33. TS\_CACHED\_BRUSH
34. PATBLT\_ORDER::BrushHatch not present
35. PATBLT\_ORDER::BrushExtra not present

### MultiPatBlt

The following is an annotated dump of a [MultiPatBlt (section 2.2.2.2.1.1.2.4)](#Section_5f677576a62e48d9a58d711490882365) Primary Drawing Order.

1. 00000000 09 10 df 31 e4 00 b7 00 0e 02 c5 01 5a ff ff 00 ...1........Z...
2. 00000010 ae b2 04 19 00 08 40 80 e4 80 b7 82 0e 1b 1b 81 ......@.........
3. 00000020 07 32 81 39 80 d5 32 fe c7 32 82 0e 81 78 .2.9..2..2...x
4. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
5. 0x09
6. = 0x01 |
7. 0x08
8. = TS\_STANDARD |
9. TS\_TYPE\_CHANGE
10. 10 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_MULTIPATBLT\_ORDER
11. df 31 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x31df
12. Binary of 0x31df = 0011 0001 1101 1111
13. Fields 1-5, 7-9, 13, 14 are present
14. e4 00 -> MULTI\_PATBLT\_ORDER::nLeftRect = 0xe4 = 228
15. b7 00 -> MULTI\_PATBLT\_ORDER::nTopRect = 0xb7 = 183
16. 0e 02 -> MULTI\_PATBLT\_ORDER::nWidth = 0x020e = 526
17. c5 01 -> MULTI\_PATBLT\_ORDER::nHeight = 0x01c5 = 453
18. 5a -> MULTI\_PATBLT\_ORDER::bRop = 0x5a = ROP Table Entry #90 = 0x005a0049
19. MULTI\_PATBLT\_ORDER::BackColor not present
20. ff ff 00 -> MULTI\_PATBLT\_ORDER::ForeColor
21. TS\_COLOR::RedOrPaletteIndex = 0xff
22. TS\_COLOR::Green = 0xff
23. TS\_COLOR::Blue = 0x00
24. ae -> MULTI\_PATBLT\_ORDER::BrushOrgX = 0xae = -82
25. b2 -> MULTI\_PATBLT\_ORDER::BrushOrgY = 0xb2 = -78
26. MULTI\_PATBLT\_ORDER:BrushStyle not present
27. MULTI\_PATBLT\_ORDER:BrushHatch not present
28. MULTI\_PATBLT\_ORDER:BrushExtra not present
29. 04 -> MULTI\_PATBLT\_ORDER::nDeltaEntries = 0x04 = 4 entries
30. 19 00 -> VARIABLE2\_FIELD::cbData = 0x19 = 25 bytes
31. 08 40 80 e4 80 b7 82 0e 1b 1b 81 07 32 81 39 80
32. d5 32 fe c7 32 82 0e 81 78 -> VARIABLE2\_FIELD::rgbData
33. 08 40 -> DELTA\_RECTS\_FIELD::zeroBits = binary:0000 1000 0100 0000
34. Rectangle #1:
35. 80 e4 -> Delta Left = 0xe4 = 228
36. 80 b7 -> Delta Top = 0xb7 = 183
37. 82 0e -> Delta Width = 0x20e = 526
38. 1b -> Delta Height = 0x1b = 27
39. Rectangle is (228, 183, 228 + 526 = 754, 183 + 27 = 210)
40. Rectangle #2:
41. Delta Left = 0 (zeroBit is 1)
42. 1b -> Delta Top = 0x1b = 27
43. 81 07 -> Delta Width = 0x107 = 263
44. 32 -> Delta Height = 0x32 = 50
45. Rectangle is (228 + 0 = 228, 183 + 27 = 210, 228 + 263 = 491, 210 + 50 = 260)
46. Rectangle #3
47. 81 39 -> Delta Left = 0x139 = 313
48. Delta Top = 0 (zeroBit is 1)
49. 80 d5 -> Delta Width = 0xd5 = 213
50. 32 -> Delta Height = 0x32 = 50
51. Rectangle is (228 + 313 = 541, 210 + 0 = 210, 541 + 213 = 754, 210 + 50 = 260)
52. Rectangle #4
53. fe c7 -> Delta Left = 0xfffffec7 = -313
54. 32 -> Delta Top = 0x32 = 50
55. 82 0e -> Delta Width = 0x20e = 526
56. 81 78 -> Delta Height = 0x178 = 376
57. Rectangle is (541 - 313 = 228, 210 + 50 = 260, 228 + 526 = 754, 260 + 376 = 636)

### OpaqueRect

The following is an annotated dump of an [OpaqueRect (section 2.2.2.2.1.1.2.5)](#Section_1eead7aaac63411a9f8cb1b227526877) Primary Drawing Order.

1. 00000000 09 0a 3c 00 04 00 03 73 02 06 ..<....s..
2. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
3. 0x09
4. = 0x01 |
5. 0x08
6. = TS\_STANDARD |
7. TS\_TYPE\_CHANGE
8. 0a -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_OPAQUERECT\_ORDER
9. 3c -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x3c
10. Binary of 0x3c = 0011 1100
11. Fields 3-6 are present
12. OPAQUERECT\_ORDER::nLeftRect not present
13. OPAQUERECT\_ORDER::nTopRect not present
14. 00 04 -> OPAQUERECT\_ORDER::nWidth = 0x0400 = 1024
15. 00 03 -> OPAQUERECT\_ORDER::nHeight = 0x0300 = 768
16. 74 -> OPAQUERECT\_ORDER::RedOrPaletteIndex = 0x74
17. 02 -> OPAQUERECT\_ORDER::Green = 0x02
18. 06 -> OPAQUERECT\_ORDER::Blue = 0x06

### MultiOpaqueRect

The following is an annotated dump of a [MultiOpaqueRect (section 2.2.2.2.1.1.2.6)](#Section_74e46cd9d6494bbe9f8af4200f23302e) Primary Drawing Order.

1. 00000000 09 12 bf 01 87 01 1c 01 f1 00 12 00 5c ef 04 16 ............\...
2. 00000010 00 08 40 81 87 81 1c 80 f1 01 01 01 10 80 f0 01 ..@.............
3. 00000020 10 ff 10 10 80 f1 01 .......
4. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
5. 0x09
6. = 0x01 |
7. 0x08
8. = TS\_STANDARD |
9. TS\_TYPE\_CHANGE
10. 12 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_MULTIOPAQUERECT\_ORDER
11. bf 01 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x01bf
12. Binary of 0x01bf = 0000 0001 1011 1111
13. Fields 1-6,8-9 are present
14. 87 01 -> MULTI\_OPAQUERECT\_ORDER::nLeftRect = 0x0187 = 391
15. 1c 01 -> MULTI\_OPAQUERECT\_ORDER::nTopRect = 0x011c = 284
16. f1 00 -> MULTI\_OPAQUERECT\_ORDER::nWidth = 0x00f1 = 241
17. 12 00 -> MULTI\_OPAQUERECT\_ORDER::nHeight = 0x0012 = 18
18. 5c -> MULTI\_OPAQUERECT\_ORDER::RedOrPaletteIndex = 0x5c
19. ef -> MULTI\_OPAQUERECT\_ORDER::Green = 0xef
20. MULTI\_OPAQUERECT\_ORDER::Blue not present
21. 04 -> MULTI\_OPAQUERECT\_ORDER::nDeltaEntries = 0x04 = 4 entries
22. 16 00 -> VARIABLE2\_FIELD::cbData = 0x16 = 22 bytes
23. 08 40 81 87 81 1c 80 f1 01 01 01 10 80 f0 01 10
24. ff 10 10 80 f1 01 -> VARIABLE2\_FIELD::rgbData
25. 08 40 -> DELTA\_RECTS\_FIELD::zeroBits = binary:0000 1000 0100 0000
26. Rectangle #1:
27. 81 87 -> Delta Left = 0x187 = 391
28. 81 1c -> Delta Top = 0x11c = 284
29. 80 f1 -> Delta Width = 0xf1 = 241
30. 01 -> Delta Height = 0x1 = 1
31. Rectangle is (391, 284, 391 + 241 = 632, 284 + 1 = 285)
32. Rectangle #2:
33. Delta Left = 0 (zeroBit is 1)
34. 01 -> Delta Top = 1
35. 01 -> Delta Width = 1
36. 10 -> Delta Height = 0x10 = 16
37. Rectangle is (391 + 0 = 391, 284 + 1 = 285, 391 + 1 = 392, 285 + 16 = 301)
38. Rectangle #3:
39. 80 f0 -> Delta Left = 0xf0 = 240
40. Delta Top = 0 (zeroBit is 1)
41. 01 -> Delta Width = 1
42. 10 -> Delta Height = 0x10 = 16
43. Rectangle is (391 + 240 = 631, 285 + 0 = 285, 631 + 1 = 632, 285 + 16 = 301)
44. Rectangle #4:
45. ff 10 -> Delta Left = 0xffffff10 = -240
46. 10 -> Delta Top = 0x10 = 16
47. 80 f1 -> Delta Width = 0xf1 = 241
48. 01 -> Delta Height = 0x1 = 1
49. Rectangle is (631 - 240 = 391, 285 + 16 = 301, 391 + 241 = 632, 301 + 1 = 302)

### ScrBlt

The following is an annotated dump of an [ScrBlt (section 2.2.2.2.1.1.2.7)](#Section_a4e322b0cd644dfc8e1af24dc0edc99d) Primary Drawing Order.

1. 00000000 09 02 7d 07 00 a1 01 f1 00 cc 2f 01 8e ..}......./..
2. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
3. 0x09
4. = 0x01 |
5. 0x08
6. = TS\_STANDARD |
7. TS\_TYPE\_CHANGE
8. 02 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_SCRBLT\_ORDER
9. 7d -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x7d
10. Binary of 0x7d = 0011 11101
11. Fields 1, 3-7 are present
12. 07 00 -> SCRBLT\_ORDER::nLeftRect = 0x07 = 7
13. SCRBLT\_ORDER::nTopRect not present
14. a1 01 -> SCRBLT\_ORDER::nWidth = 0x01a1 = 417
15. f1 00 -> SCRBLT\_ORDER::nHeight = 0x00f1 = 241
16. cc -> SCRBLT\_ORDER::bRop = 0xcc = ROP Table Entry #204 = 0x00cc0020
17. 2f 01 -> SCRBLT\_ORDER::nXSrc = 0x012f = 303
18. 8e 00 -> SCRBLT\_ORDER::nYSrc = 0x008e = 142

### MultiScrBlt

The following is an annotated dump of a [MultiScrBlt (section 2.2.2.2.1.1.2.8)](#Section_31b79b4915b74709bbc2d58ce0362292) Primary Drawing Order.

1. 00000000 09 11 ff 01 2d 03 4e 01 10 00 10 00 cc 2d 03 05 ....-.N......-..
2. 00000010 01 10 4a 00 00 00 00 00 00 00 00 00 83 2d 81 5d ..J..........-.]
3. 00000020 10 01 01 7f 0f 01 01 7f 0e 01 01 7f 0d 01 01 7f ................
4. 00000030 0c 01 01 7f 0b 01 01 7f 0a 01 01 7f 09 01 01 7f ................
5. 00000040 08 01 01 7f 07 01 01 7f 06 01 01 7f 05 01 01 7f ................
6. 00000050 04 01 01 7f 03 01 01 7f 02 01 01 7f 01 01 ..............
7. 09 -> MULTI\_SCRBLT\_ORDER::controlFlags = 0x09
8. 0x09
9. = 0x01 |
10. 0x08
11. = TS\_STANDARD |
12. TS\_TYPE\_CHANGE
13. 11 -> MULTI\_SCRBLT\_ORDER::orderType = TS\_ENC\_MULTISCRBLT\_ORDER
14. ff 01 -> MULTI\_SCRBLT\_ORDER::fieldFlags = 0x01ff
15. Binary of 0x01ff = 0000 0001 1111 1111
16. Fields 1-9 are present
17. 2d 03 -> MULTI\_SCRBLT\_ORDER::nLeftRect = 0x032d = 813
18. 4e 01 -> MULTI\_SCRBLT\_ORDER::nTopRect = 0x014e = 334
19. 10 00 -> MULTI\_SCRBLT\_ORDER::nWidth = 0x10 = 16
20. 10 00 -> MULTI\_SCRBLT\_ORDER::nHeight = 0x10 = 16
21. cc -> MULTI\_SCRBLT\_ORDER::bRop = 0xcc = ROP Table Entry #204 = 0x00cc0020
22. 2d 03 -> MULTI\_SCRBLT\_ORDER::nXSrc = 0x032d = 813
23. 05 01 -> MULTI\_SCRBLT\_ORDER::nYSrc = 0x0105 = 261
24. 10 -> MULTI\_SCRBLT\_ORDER::nDeltaEntries = 0x10 = 16 entries
25. 4a 00 -> VARIABLE2\_FIELD::cbData = 0x4a = 74 bytes
26. 00 00 00 00 00 00 00 00 83 2d 81 5d 10 01 01 7f
27. 0f 01 01 7f 0e 01 01 7f 0d 01 01 7f 0c 01 01 7f
28. 0b 01 01 7f 0a 01 01 7f 09 01 01 7f 08 01 01 7f
29. 07 01 01 7f 06 01 01 7f 05 01 01 7f 04 01 01 7f
30. 03 01 01 7f 02 01 01 7f 01 01 -> VARIABLE2\_FIELD::rgbData
31. 00 00 00 00 00 00 00 00 -> DELTA\_RECTS\_FIELD::zeroBits = 0
32. Rectangle #1:
33. 83 2d -> Delta Left -> 0x32d = 813
34. 81 5d -> Delta Top -> 0x15d = 349
35. 10 -> Delta Width -> 0x10 = 16
36. 01 -> Delta Height -> 0x1 = 1
37. Rectangle is (813, 349, 813 + 16 = 829, 349 + 1 = 350)
38. Rectangle #2
39. 01 -> Delta Left -> 0x1 = 1
40. 7f -> Delta Top -> 0x7f = -1
41. 0f -> Delta Width -> 0xf = 15
42. 01 -> Delta Height -> 0x1 = 1
43. Rectangle is (813 + 1 = 814, 349 - 1 = 348, 814 + 15 = 829, 348 + 1 = 349)
44. Rectangle #3
45. 01 -> Delta Left -> 0x1 = 1
46. 7f -> Delta Top -> 0x7f = -1
47. 0e -> Delta Width -> 0xe = 14
48. 01 -> Delta Height -> 0x1 = 1
49. Rectangle is (815, 347, 829, 348)
50. Rectangle #4
51. 01 -> Delta Left -> 0x1 = 1
52. 7f -> Delta Top -> 0x7f = -1
53. 0d -> Delta Width -> 0xd = 13
54. 01 -> Delta Height -> 0x1 = 1
55. Rectangle is (816, 346, 829, 347)
56. Rectangle #5
57. 01 -> Delta Left -> 0x1 = 1
58. 7f -> Delta Top -> 0x7f = -1
59. 0c -> Delta Width -> 0xc = 12
60. 01 -> Delta Height -> 0x1 = 1
61. Rectangle is (817, 345, 829, 346)
62. Rectangle #6
63. 01 -> Delta Left -> 0x1 = 1
64. 7f -> Delta Top -> 0x7f = -1
65. 0b -> Delta Width -> 0xb = 11
66. 01 -> Delta Height -> 0x1 = 1
67. Rectangle is (818, 344, 829, 345)
68. Rectangle #7
69. 01 -> Delta Left -> 0x1 = 1
70. 7f -> Delta Top -> 0x7f = -1
71. 0a -> Delta Width -> 0xa = 10
72. 01 -> Delta Height -> 0x1 = 1
73. Rectangle is (819, 343, 829, 344)
74. Rectangle #8
75. 01 -> Delta Left -> 0x1 = 1
76. 7f -> Delta Top -> 0x7f = -1
77. 09 -> Delta Width -> 0x9 = 9
78. 01 -> Delta Height -> 0x1 = 1
79. Rectangle is (820, 342, 829, 343)
80. Rectangle #9
81. 01 -> Delta Left -> 0x1 = 1
82. 7f -> Delta Top -> 0x7f = -1
83. 08 -> Delta Width -> 0x8 = 8
84. 01 -> Delta Height -> 0x1 = 1
85. Rectangle is (821, 341, 829, 342)
86. Rectangle #10
87. 01 -> Delta Left -> 0x1 = 1
88. 7f -> Delta Top -> 0x7f = -1
89. 07 -> Delta Width -> 0x7 = 7
90. 01 -> Delta Height -> 0x1 = 1
91. Rectangle is (822, 340, 829, 341)
92. Rectangle #11
93. 01 -> Delta Left -> 0x1 = 1
94. 7f -> Delta Top -> 0x7f = -1
95. 06 -> Delta Width -> 0x6 = 6
96. 01 -> Delta Height -> 0x1 = 1
97. Rectangle is (823, 339, 829, 340)
98. Rectangle #12
99. 01 -> Delta Left -> 0x1 = 1
100. 7f -> Delta Top -> 0x7f = -1
101. 05 -> Delta Width -> 0x5 = 5
102. 01 -> Delta Height -> 0x1 = 1
103. Rectangle is (824, 338, 829, 339)
104. Rectangle #13
105. 01 -> Delta Left -> 0x1 = 1
106. 7f -> Delta Top -> 0x7f = -1
107. 04 -> Delta Width -> 0x4 = 4
108. 01 -> Delta Height -> 0x1 = 1
109. Rectangle is (825, 337, 829, 338)
110. Rectangle #14
111. 01 -> Delta Left -> 0x1 = 1
112. 7f -> Delta Top -> 0x7f = -1
113. 03 -> Delta Width -> 0x3 = 3
114. 01 -> Delta Height -> 0x1 = 1
115. Rectangle is (826, 336, 829, 337)
116. Rectangle #15
117. 01 -> Delta Left -> 0x1 = 1
118. 7f -> Delta Top -> 0x7f = -1
119. 02 -> Delta Width -> 0x2 = 2
120. 01 -> Delta Height -> 0x1 = 1
121. Rectangle is (827, 335, 829, 336)
122. Rectangle #16
123. 01 -> Delta Left -> 0x1 = 1
124. 7f -> Delta Top -> 0x7f = -1
125. 01 -> Delta Width -> 0x1 = 1
126. 01 -> Delta Height -> 0x1 = 1
127. Rectangle is (828, 334, 829, 335)

### MemBlt

The following is an annotated dump of a [MemBlt (section 2.2.2.2.1.1.2.9)](#Section_84c2ec2ff776405b9b486894a28b1b14) Primary Drawing Order.

1. 00000000 49 0d 97 ff 00 02 00 06 01 15 00 15 00 I............
2. 49 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x19
3. 0x49
4. = 0x01 |
5. 0x08 |
6. 0x40
7. = TS\_STANDARD |
8. TS\_TYPE\_CHANGE |
9. TS\_ZERO\_FIELD\_BYTE\_BIT0
10. 0d -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_MEMBLT\_R2\_ORDER
11. 97 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x97
12. TS\_ZERO\_FIELD\_BYTE\_BIT0 Indicates that last field byte is not present
13. PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x0097
14. Binary of 0x0097 = 0000 0000 1001 0111
15. Fields 1-3, 5, 8 are present
16. ff 00 -> MEMBLT\_ORDER::cacheId = 0x00ff
17. Color Table Cache Index = 0x00
18. Bitmap Cache ID = 0xff (TS\_BITMAPCACHE\_SCREEN\_ID)
19. 02 00 -> MEMBLT\_ORDER::nLeftRect = 0x0002 = 2
20. 06 01 -> MEMBLT\_ORDER::nTopRect = 0x0106 = 262
21. MEMBLT\_ORDER::nWidth not present
22. 15 00 -> MEMBLT\_ORDER::nHeight = 0x015 = 21
23. MEMBLT\_ORDER::bRop not present
24. MEMBLT\_ORDER::nXSrc not present
25. 15 00 -> MEMBLT\_ORDER::nYSrc = 0x015 = 21
26. MEMBLT\_ORDER::cacheIndex not present

### Mem3Blt

The following is an annotated dump of a [Mem3Blt (section 2.2.2.2.1.1.2.10)](#Section_3aa21a4a031c4a3f8fc5159068f30237) Primary Drawing Order.

1. 00000000 49 0e 3f 8e 01 00 3c 01 1f 01 40 00 0c 00 b8 ff I.?...<...@.....
2. 00000010 ff 00 3b 1e ff 7f 03 71 00 21 04 05 40 0d 40 73 ..;....q.!..@.@s
3. 00000020 07 20 1c 84 ff ff ff ff ff ff ff ff 00 21 fe 44 . ...........!.D
4. 00000030 e7 38 81 c8 86 00 00 00 00 ff ff 00 00 ff ff ff .8..............
5. 00000040 ff 43 c9 11 00 26 01 fd 43 f7 01 80 fd 5d 03 f0 .C...&..C....]..
6. 00000050 01 00 00 81 00 00 00 00 09 00 c0 f3 00 c0 03 e0 ................
7. 00000060 01 00 80 70 04 80 03 e0 08 00 c0 84 00 00 00 00 ...p............
8. 00000070 ff ff ff ff 43 cf e0 79 01 26 40 20 3e 00 00 70 ....C..y.&@ >..p
9. 49 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x19
10. 0x49
11. = 0x01 |
12. 0x08 |
13. 0x40
14. = TS\_STANDARD |
15. TS\_TYPE\_CHANGE |
16. TS\_ZERO\_FIELD\_BYTE\_BIT0
17. 0e -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_MEM3BLT\_R2\_ORDER
18. 3f 8e -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x8e3f
19. TS\_ZERO\_FIELD\_BYTE\_BIT0 Indicates that last field byte is 0
20. PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x008e3f
21. Binary of 0x008e3f = 0000 0000 1000 1110 0011 1111
22. Fields 1-6, 10-12, 16 are present
23. 01 00 -> MEM3BLT\_ORDER::cacheId = 0x0001
24. Color Table Cache Index = 0x00
25. Bitmap Cache ID = 0x01
26. 3c 01 -> MEM3BLT\_ORDER::nLeftRect = 0x013c = 316
27. 1f 01 -> MEM3BLT\_ORDER::nTopRect = 0x011f = 287
28. 40 00 -> MEM3BLT\_ORDER::nWidth = 0x0040 = 64
29. 0c 00 -> MEM3BLT\_ORDER::nHeight = 0x015 = 12
30. b8 -> MEM3BLT\_ORDER::bRop = 0xb8 = ROP Table Entry #184 = 0x00b80666
31. MEM3BLT\_ORDER::nXSrc not present
32. MEM3BLT\_ORDER::nYSrc not present
33. MEM3BLT\_ORDER::BackColor not present
34. ff ff 00 -> MEM3BLT\_ORDER::ForeColor
35. TS\_COLOR::RedOrPaletteIndex = 0xff
36. TS\_COLOR::Green = 0xff
37. TS\_COLOR::Blue = 0x00
38. 3b -> MEM3BLT\_ORDER::BrushOrgX = 0x3b = 59
39. 1e -> MEM3BLT\_ORDER::BrushOrgY = 0x1e = 30
40. MEM3BLT\_ORDER::BrushStyle not present
41. MEM3BLT\_ORDER::BrushHatch not present
42. MEM3BLT\_ORDER::BrushExtra not present
43. ff 7f -> MEM3BLT\_ORDER::cacheIndex = 0x7fff = 32767

### LineTo

The following is an annotated dump of a [LineTo (section 2.2.2.2.1.1.2.11)](#Section_27276fcf32be49a99a1797ec6167ac7e) Primary Drawing Order.

1. 00000000 1d 09 1e 02 a5 3d 03 1a 4a 03 1b 03 b1 0e a6 5b .....=..J......[
2. 00000010 ef 00 ..
3. 1d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x1d
4. 0x1d
5. = 0x01 |
6. 0x04 |
7. 0x08 |
8. 0x10
9. = TS\_STANDARD |
10. TS\_BOUNDS |
11. TS\_TYPE\_CHANGE |
12. TS\_DELTA\_COORDINATES
13. 09 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_LINETO\_ORDER
14. 1e 02 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x021e
15. Binary of 0x021e = 0000 0010 0001 1110
16. a5 -> PRIMARY\_DRAWING\_ORDER::bounds::flags = 0xa5
17. 0xa5
18. = 0x01 |
19. 0x04 |
20. 0x20 |
21. 0x80
22. = TS\_BOUND\_LEFT |
23. TS\_BOUND\_RIGHT |
24. TS\_BOUND\_DELTA\_TOP |
25. TS\_BOUND\_DELTA\_BOTTOM
26. 3d 03 -> PRIMARY\_DRAWING\_ORDER::bounds::left = 0x033d = 829
27. 1a -> PRIMARY\_DRAWING\_ORDER::bounds::top (delta) = 0x1a = 26 pixels from last top bounds
28. PRIMARY\_DRAWING\_ORDER::bounds::top = last bounds::top (0xe7) + 0x1a = 0x101 = 257
29. 4a 03 -> PRIMARY\_DRAWING\_ORDER::bounds::right = 0x034a = 842
30. 1b -> PRIMARY\_DRAWING\_ORDER::bounds::bottom (delta) = 0x1b = 27 pixels from last bottom bounds
31. PRIMARY\_DRAWING\_ORDER::bounds::bottom = last bounds::bottom (0xf3) + 0x1b = 0x10e = 270
32. LINETO\_ORDER::BackMode not present
33. 03 -> LINETO\_ORDER::nXStart(delta) = 0x03 = 3 pixels from last LINETO\_ORDER::nXStart
34. LINETO\_ORDER::nXStart = last LINETO\_ORDER::nXStart (0x33a) + 0x03 = 0x33d = 829
35. b1 -> LINETO\_ORDER::nYStart(delta) = 0xb1 = -79 pixels from last LINETO\_ORDER::nYStart
36. LINETO\_ORDER::nYStart = last LINETO\_ORDER::nYStart (0x15e) + 0xb1 = 0x10f = 271
37. 0e -> LINETO\_ORDER::nXEnd(delta) = 0x0e = 14 pixels from last LINETO\_ORDER::nXEnd
38. LINETO\_ORDER::nXEnd = last LINETO\_ORDER::nXEnd (0x33d) + 0x0e = 0x34b = 843
39. a6 -> LINETO\_ORDER::nYEnd(delta) = 0xa6 = -90 pixels from last LINETO\_ORDER::nYEnd
40. LINETO\_ORDER::nYEnd = last LINETO\_ORDER::nYEnd (0x15b) + 0xa6 = 0x101 = 257
41. LINETO\_ORDER::BackColor not present
42. LINETO\_ORDER::bRop2 not present
43. LINETO\_ORDER::PenStyle not present
44. LINETO\_ORDER::PenWidth not present
45. 5b ef 00 -> LINETO\_ORDER::PenColor
46. TS\_COLOR::RedOrPaletteIndex = 0x5b
47. TS\_COLOR::Green = 0xef
48. TS\_COLOR::Blue = 0x00

### SaveBitmap

The following is an annotated dump of a [SaveBitmap (section 2.2.2.2.1.1.2.12)](#Section_db4ce8dbcf76483d99ba354cd010ea4a) Primary Drawing Order.

1. 00000000 19 0b 1b 74 45 00 00 79 70 10 ...tE..yp.
2. 19 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
3. 0x19
4. = 0x01 |
5. 0x08 |
6. 0x10
7. = TS\_STANDARD |
8. TS\_TYPE\_CHANGE |
9. TS\_DELTA\_COORDINATES
10. 0b -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_SAVEBITMAP\_ORDER
11. 1b -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x1b
12. Binary of 0x1e = 0001 1011
13. Fields 1-2, 4-5 are present
14. 74 45 00 00 -> SAVEBITMAP\_ORDER::SavedBitmapPosition = 0x4574 = 17780
15. Y granularity = 20
16. Y position = (17780 / (480 \* 20)) \* 20 = (17780 / 9600) \* 20 = 1 \* 20 = 20
17. X position = (17780 - (20 \* 480)) / 20 = (17780 - 9600) / 20 = 8180 / 20 = 409
18. 79 -> SAVEBITMAP\_ORDER::nLeftRect (delta) = 0x79 = 121
19. SAVEBITMAP\_ORDER::nTopRect (delta) not present
20. 70 -> SAVEBITMAP\_ORDER::nRightRect (delta) = 0x70 = 112
21. 10 -> SAVEBITMAP\_ORDER::nBottomRect (delta) = 0x10 = 16
22. SAVEBITMAP\_ORDER::Operation is not present

### GlyphIndex

The following sections provide examples of annotated dumps of [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) Primary Drawing Orders.

#### Example 1

The following is the first example of an annotated dump of a [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) Primary Drawing Order.

1. 00000000 19 1b 00 01 20 6a 02 27 38 00 39 07 3a 06 3b 07 .... j.'8.9.:.;.
2. 00000010 3c 06 3d 06 18 04 1f 06 17 02 14 04 1b 06 19 06 <.=.............
3. 00000020 45 05 18 06 1f 06 1f 02 14 02 46 06 ff 15 24 E.........F...$
4. 19 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x19
5. 0x09
6. = 0x01 |
7. 0x08 |
8. 0x10
9. = TS\_STANDARD |
10. TS\_TYPE\_CHANGE |
11. TS\_DELTA\_COORDINATES
12. 1b -> PRIMARY\_DRAWING\_ORDER::orderType = 0x1b = TS\_ENC\_INDEX\_ORDER
13. 00 01 20 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x200100
14. Binary of 0x200100 = 10 0000 0000 0001 0000 0000
15. Fields 9, 22 present
16. 6a 02 -> GLYPHINDEX\_ORDER::BkRight = 0x026a = 618
17. 27 -> VARIABLE1\_FIELD::cbData = 0x27 = 39 bytes
18. 38 00 -> Glyph Cache Index = 56, Delta = 0
19. 39 07 -> Glyph Cache Index = 57, Delta = 7
20. 3a 06 -> Glyph Cache Index = 58, Delta = 6
21. 3b 07 -> Glyph Cache Index = 59, Delta = 7
22. 3c 06 -> Glyph Cache Index = 60, Delta = 6
23. 3d 06 -> Glyph Cache Index = 61, Delta = 6
24. 18 04 -> Glyph Cache Index = 24, Delta = 4
25. 1f 06 -> Glyph Cache Index = 31, Delta = 6
26. 17 02 -> Glyph Cache Index = 23, Delta = 2
27. 14 04 -> Glyph Cache Index = 20, Delta = 4
28. 1b 06 -> Glyph Cache Index = 27, Delta = 6
29. 19 06 -> Glyph Cache Index = 25, Delta = 6
30. 45 05 -> Glyph Cache Index = 69, Delta = 5
31. 18 06 -> Glyph Cache Index = 24, Delta = 6
32. 1f 06 -> Glyph Cache Index = 31, Delta = 6
33. 1f 02 -> Glyph Cache Index = 31, Delta = 2
34. 14 02 -> Glyph Cache Index = 20, Delta = 2
35. 46 06 -> Glyph Cache Index = 70, Delta = 6
36. ff -> ADD Operation
37. 15 -> Fragment Cache Index = 0x15 = 21
38. 24 -> Size of Fragment = 0x24 = 36 bytes

#### Example 2

The following is the second example of an annotated dump of a [GlyphIndex (section 2.2.2.2.1.1.2.13)](#Section_8e906465a1da4d0691db54beb7827381) Primary Drawing Order.

1. 00000000 19 1b e8 3f 38 00 ff ff ff 0c 02 6e 01 4d 02 7b ...?8......n.M.{
2. 00000010 01 09 02 6e 01 f6 02 7b 01 0c 02 79 01 03 fe 04 ...n...{...y....
3. 00000020 00 .
4. 19 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x19
5. 0x09
6. = 0x01 |
7. 0x08 |
8. 0x10
9. = TS\_STANDARD |
10. TS\_TYPE\_CHANGE |
11. TS\_DELTA\_COORDINATES
13. 1b -> PRIMARY\_DRAWING\_ORDER::orderType = 0x1b = TS\_ENC\_INDEX\_ORDER
14. e8 3f 38 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x383fe8
15. Binary of 0x383fe8 = 11 1000 0011 1111 1110 1000
16. Fields 4, 6-14, 20-22 are present
17. 00 -> GLYPHINDEX\_ORDER::fOpRedundant = 0x00 = FALSE
18. ff ff ff -> GLYPHINDEX\_ORDER::ForeColor
19. TS\_COLOR::RedOrPaletteIndex = 0xff
20. TS\_COLOR::Green = 0xff
21. TS\_COLOR::Blue = 0xff
22. 0c 02 -> GLYPHINDEX\_ORDER::BkLeft = 0x020c = 524
23. 6e 01 -> GLYPHINDEX\_ORDER::BkTop = 0x016e = 366
24. 4d 02 -> GLYPHINDEX\_ORDER::BkRight = 0x024d = 589
25. 7b 01 -> GLYPHINDEX\_ORDER::BkBottom = 0x017b = 379
26. 09 02 -> GLYPHINDEX\_ORDER::OpLeft = 0x0209 = 521
27. 6e 01 -> GLYPHINDEX\_ORDER::OpTop = 0x016e = 366
28. f6 02 -> GLYPHINDEX\_ORDER::OpRight = 0x02f6 = 758
29. 7b 01 -> GLYPHINDEX\_ORDER::OpBottom = 0x017b = 379
30. 0c 02 -> GLYPHINDEX\_ORDER::X = 0x020c = 524
31. 79 01 -> GLYPHINDEX\_ORDER::Y = 0x0179 = 377
32. 03 -> VARIABLE1\_FIELD::cbData = 0x03 = 3 bytes
33. fe 04 00 -> VARIABLE1\_FIELD::rgbData
34. fe -> USE Operation
35. 04 -> Fragment Cache Index = 0x04 = 4
36. 00 -> Delta

### FastIndex

The following sections provide examples of annotated dumps of [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7) Primary Drawing Orders.

#### Example 1

The following is the first example of an annotated dump of a [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7) Primary Drawing Order.

1. 00000000 09 13 ff 70 07 00 03 ff ff 00 74 3b 00 0e 00 71 ...p......t;...q
2. 00000010 00 42 00 7e 00 00 80 7c 00 15 00 00 01 06 02 04 .B.~...|........
3. 00000020 03 08 05 09 06 06 06 06 07 06 08 02 ff 00 12 ...............
4. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
5. 0x09
6. = 0x01 |
7. 0x08
8. = TS\_STANDARD |
9. TS\_TYPE\_CHANGE
10. 13 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x13 = TS\_ENC\_FAST\_INDEX\_ORDER
11. ff 70 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x70ff
12. Binary of 0x70ff = 0111 0000 1111 1111
13. Fields 1-8, 13-15 present
14. 07 -> FASTINDEX\_ORDER::cacheId = 0x07
15. 00 03 -> FASTINDEX\_ORDER::fDrawing = 0x0300
16. fAccel = 0x03
17. ulCharInc = 0x00
18. ff ff 00 -> FASTINDEX\_ORDER::BackColor
19. TS\_COLOR::RedOrPaletteIndex = 0xff
20. TS\_COLOR::Green = 0xff
21. TS\_COLOR::Blue = 0x00
22. 74 3b 00 -> FASTINDEX\_ORDER::ForeColor
23. TS\_COLOR::RedOrPaletteIndex = 0x74
24. TS\_COLOR::Green = 0x3b
25. TS\_COLOR::Blue = 0x00
26. 0e 00 -> FASTINDEX\_ORDER::BkLeft = 0xe = 14
27. 71 00 -> FASTINDEX\_ORDER::BkTop = 0x71 = 113
28. 42 00 -> FASTINDEX\_ORDER::BkRight = 0x42 = 66
29. 7e 00 -> FASTINDEX\_ORDER::BkBottom = 0x7e = 126
30. FASTINDEX\_ORDER::OpLeft not present
31. FASTINDEX\_ORDER::OpTop not present
32. FASTINDEX\_ORDER::OpRight not present
33. FASTINDEX\_ORDER::OpBottom not present
34. 00 80 -> FASTINDEX\_ORDER::X = 0x8000 = -32768
35. 7c 00 -> FASTINDEX\_ORDER::Y = 0x7c = 124
36. 15 -> VARIABLE1\_FIELD::cbData = 0x15 = 21 bytes
37. 00 00 01 06 02 04 03 08 05 09 06 06 06 06 07 06
38. 08 02 ff 00 12 -> VARIABLE1\_FIELD::rgbData
39. 00 00 -> Glyph Cache Index = 0, Delta = 0
40. 01 06 -> Glyph Cache Index = 1, Delta = 6
41. 02 04 -> Glyph Cache Index = 2, Delta = 4
42. 03 08 -> Glyph Cache Index = 3, Delta = 8
43. 05 09 -> Glyph Cache Index = 4, Delta = 9
44. 06 06 -> Glyph Cache Index = 5, Delta = 6
45. 06 06 -> Glyph Cache Index = 6, Delta = 6
46. 07 06 -> Glyph Cache Index = 7, Delta = 6
47. 08 02 -> Glyph Cache Index = 8, Delta = 2
48. ff -> ADD Operation
49. 00 -> Fragment Cache Index = 0
50. 12 -> Size of Fragment = 0x12 = 18 bytes

#### Example 2

The following is the second example of an annotated dump of a [FastIndex (section 2.2.2.2.1.1.2.14)](#Section_ac55f919acb943c1a3ec33802929f9f7) Primary Drawing Order.

1. 00000000 19 13 e0 60 e2 d5 e2 e2 03 fe 1c 00 ...`........
2. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
3. 0x19
4. = 0x01 |
5. 0x08 |
6. 0x10
7. = TS\_STANDARD |
8. TS\_TYPE\_CHANGE |
9. TS\_DELTA\_COORDINATES
10. 13 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x13 = TS\_ENC\_FAST\_INDEX\_ORDER
11. e0 60 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x60e0
12. Binary of 0x60e0 = 0110 0000 1110 0000
13. Fields 6-8, 14-15 are present
14. FASTINDEX\_ORDER::cacheId not present
15. FASTINDEX\_ORDER::fDrawing not present
16. FASTINDEX\_ORDER::BackColor not present
17. FASTINDEX\_ORDER::ForeColor not present
18. FASTINDEX\_ORDER::BkLeft not present
19. e2 -> FASTINDEX\_ORDER::BkTop(delta) = 0xe2 = -30 pixels from last FASTINDEX\_ORDER::BkTop
20. FASTINDEX\_ORDER::BkTop = last FASTINDEX\_ORDER::BkTop (0x159) + 0xe2 = 0x13b = 315
21. d5 -> FASTINDEX\_ORDER::BkRight(delta) = 0xd5 = -43 pixels from last FASTINDEX\_ORDER::BkRight
22. FASTINDEX\_ORDER::BkRight = last FASTINDEX\_ORDER::BkRight (0x161) + 0xd5 = 0x136 = 310
23. e2 -> FASTINDEX\_ORDER::BkBottom(delta) = 0xe2 = -30 pixels from last FASTINDEX\_ORDER::BkBottom
24. FASTINDEX\_ORDER::BkBottom = last FASTINDEX\_ORDER::BkBottom (0x166) + 0xe2 = 0x148 = 328
25. FASTINDEX\_ORDER::OpLeft not present
26. FASTINDEX\_ORDER::OpTop not present
27. FASTINDEX\_ORDER::OpRight not present
28. FASTINDEX\_ORDER::OpBottom not present
29. FASTINDEX\_ORDER::X not present
30. e2 -> FASTINDEX\_ORDER::Y(delta) = 0xe2 = -30 pixels from last FASTINDEX\_ORDER::Y
31. FASTINDEX\_ORDER::Y = last FASTINDEX\_ORDER::Y(0x164) + 0xe2 = 0x146 = 326
32. 03 -> VARIABLE1\_FIELD::cbData = 0x03 = 3 bytes
33. fe 1c 00 -> VARIABLE1\_FIELD::rgbData
34. fe -> USE Operation
35. 1c -> Fragment Cache Index = 0x1c = 28
36. 00 -> Delta

### FastGlyph

The following is an annotated dump of a [FastGlyph (section 2.2.2.2.1.1.2.15)](#Section_839e963ce24a42adb4bdeb108340629c) Primary Drawing Order.

1. 00000000 0d 18 fb 7e 0f 8b 00 b1 00 2c 04 bd 00 06 00 03 ...~.....,......
2. 00000010 ff ff 00 8b 00 b1 00 93 00 be 00 0d 00 fe 7f 00 ................
3. 00000020 80 00 80 bb 00 13 00 01 4a 06 0a 80 80 80 b8 c4 ........J.......
4. 00000030 84 84 84 84 84 00 00 68 00 .......h.
5. 0d -> PRIMARY\_DRAWING\_ORDER::controlFlags
6. 0x0d
7. = 0x80 | 0x04 | 0x01
8. = TS\_TYPE\_CHANGE | TS\_BOUNDS | TS\_STANDARD
9. 18 -> PRIMARY\_DRAWING\_ORDER::orderorderType = TS\_ENC\_FAST\_GLYPH\_ORDER (0x18)
10. fb 7e -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x7efb
11. Binary of 0x7efb = 0111 1110 1111 1011
12. Fields 1-2, 4-8, 10-15 are present
13. 0f -> PRIMARY\_DRAWING\_ORDER::bounds (description flag)
14. 0x0f
15. = 0x08 |
16. 0x04 |
17. 0x02 |
18. 0x01
19. = TS\_BOUND\_BOTTOM |
20. TS\_BOUND\_RIGHT |
21. TS\_BOUND\_TOP |
22. TS\_BOUND\_LEFT
23. 8b 00 -> PRIMARY\_DRAWING\_ORDER::bounds::left = 0x008b = 139
24. b1 00 -> PRIMARY\_DRAWING\_ORDER::bounds::top = 0x00b1 = 177
25. 2c 04 -> PRIMARY\_DRAWING\_ORDER::bounds::right = 0x042c = 1068
26. bd 00 -> PRIMARY\_DRAWING\_ORDER::bounds::bottom = 0x00bd = 189
27. 06 -> FASTGLYPH\_ORDER::cacheId = 6
28. 00 03 -> FASTGLYPH\_ORDER::fDrawing = 0x0300
29. flAccel = 0x03 = (SO\_HORIZONTAL | SO\_FLAG\_DEFAULT\_PLACEMENT)
30. ulCharInc = 0
31. FASTGLYPH\_ORDER::BackColor not present
32. ff ff 00 -> FASTGLYPH\_ORDER::ForeColor
33. TS\_COLOR::RedOrPaletteIndex = 0xff
34. TS\_COLOR::Green = 0xff
35. TS\_COLOR::Blue = 0x00
36. 8b 00 -> FASTGLYPH\_ORDER::BkLeft = 139
37. b1 00 -> FASTGLYPH\_ORDER::BkTop = 177
38. 93 00 -> FASTGLYPH\_ORDER::BkRight = 147
39. be 00 -> FASTGLYPH\_ORDER::BkBottom = 190
40. FASTGLYPH\_ORDER::OpLeft not present
41. 0d 00 -> FASTGLYPH\_ORDER::OpTop = 0x0D = 13
42. fe 7f -> FASTGLYPH\_ORDER::OpRight = 0x7ffe = 32766
43. 00 80 -> FASTGLYPH\_ORDER::OpBottom = -32768
44. OpBottom = -32768, so OpTop (0x0D) contains encoding flags in the low 4 bits.
45. 0x0D
46. = 0x08 | 0x04 | 0x01
47. = OPRECT\_LEFT\_ABSENT | OPRECT\_TOP\_ABSENT | OPRECT\_BOTTOM\_ABSENT
48. Hence, the left, top, and bottom coordinates of the opaque rectangle all match the background text rectangle and are absent. The actual value of the right coordinate is present.
49. OpLeft = BkLeft = 139
50. OpTop = BkTop = 177
51. OpBottom = BkBottom = 190
52. OpRight = 32766
53. 00 80 -> FASTGLYPH\_ORDER::X = -32768
54. bb 00 -> FASTGLYPH\_ORDER::Y = 187
55. 13 -> VARIABLE1\_FIELD::cbData = 0x13 = 19 bytes
56. 00 01 4a 06 0a 80 80 80 b8 c4 84 84 84 84 84 00
57. 00 68 00 -> VARIABLE1\_FIELD::rgbData
58. 00 -> TS\_CACHE\_GLYPH\_DATA\_REV2::cacheIndex = 0
59. 01 -> TS\_CACHE\_GLYPH\_DATA\_REV2::x = 1
60. 4a -> TS\_CACHE\_GLYPH\_DATA\_REV2::y = -10
61. 06 -> TS\_CACHE\_GLYPH\_DATA\_REV2::cx = 6
62. 0a -> TS\_CACHE\_GLYPH\_DATA\_REV2::cy = 10
63. 80 80 80 b8 c4 84 84 84 84 84 00 00 -> TS\_CACHE\_GLYPH\_DATA\_REV2::aj
64. 0x80 -> X.....
65. 0x80 -> X.....
66. 0x80 -> X.....
67. 0xb8 -> X XXX.
68. 0xc4 -> XX...X
69. 0x84 -> X....X
70. 0x84 -> X....X
71. 0x84 -> X....X
72. 0x84 -> X....X
73. 0x84 -> X....X
74. 00 00 -> padding
75. 68 00 -> Unicode = 0x68 = 104 = h

### PolygonSC

The following is an annotated dump of a [PolygonSC (section 2.2.2.2.1.1.2.16)](#Section_22e1117fd40d41df970959389ac1f8dc) Primary Drawing Order.

1. 00000000 0d 14 7f 43 ae 00 b2 00 5c 36 02 0d 03 0d 01 20 ...C....\6.....
2. 00000010 ba 00 36 d0 00 00 00 00 00 00 00 00 00 00 00 00 ..6.............
3. 00000020 00 00 80 c0 76 80 b5 4a 80 9f ff a1 80 81 ff 7d ....v..J.......}
4. 00000030 80 5d ff 5f 33 ff 4c 08 ff 3f 5c ff 3e ff b1 ff .].\_3.L..?\.>...
5. 00000040 46 ff 8a ff 59 ff 6b ff 73 ff 52 ff 96 ff 42 ff F...Y.k.s.R...B.
6. 00000050 be ff 3e 69 ff 41 15 ff 51 80 40 ff 68 80 69 ff ..>i.A..Q.@.h.i.
7. 00000060 87 80 8b ff ad 80 a7 58 80 b9 03 80 c2 2f 80 c1 .......X...../..
8. 00000070 80 59 80 b5 80 7e 80 a1 80 9d 80 85 80 b1 80 60 .Y...~.........`
9. 00000080 80 c0 38 80 c3 0c 80 bb 60 80 ab ff b5 80 91 ff ..8.....`.......
10. 00000090 8e 80 70 ff 6e 80 49 ff 54 1e ff 44 71 ff 3d 46 ..p.n.I.T..Dq.=F
11. 000000a0 ff 41 ff 9d ff 4f ff 7a ff 65 ff 5d ff 83 ff 49 .A...O.z.e.]...I
12. 000000b0 ff aa ff 3f 53 ff 3e 7f ff 49 2b ff 5b 80 55 ff ...?S.>..I+.[.U.
13. 000000c0 76 80 7b ff 9a 80 99 41 80 b1 6e 80 bf 19 80 c3 v.{....A..n.....
14. 000000d0 80 45 80 bc 80 6c 80 ad 80 8e 80 94 80 a8 80 73 .E...l.........s
15. 000000e0 81 7d 80 6f .}.o
16. 0d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x0d
17. 0x0d
18. = 0x01 |
19. 0X04 |
20. 0x08
21. = TS\_STANDARD |
22. TS\_BOUNDS |
23. TS\_TYPE\_CHANGE
24. 14 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x14 = TS\_ENC\_POLYGON\_SC\_ORDER
25. 7f -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x7f
26. Binary of 0x7f = 0111 1111
27. Fields 1-7 are present
28. 43 -> PRIMARY\_DRAWING\_ORDER::bounds::flags = 0x43
29. 0x43
30. = 0x01 |
31. 0x02 |
32. 0x40
33. = TS\_BOUND\_LEFT |
34. TS\_BOUND\_TOP |
35. TS\_BOUND\_DELTA\_RIGHT
36. ae 00 -> PRIMARY\_DRAWING\_ORDER::bounds::left = 0x00ae = 174
37. b2 00 -> PRIMARY\_DRAWING\_ORDER::bounds::top = 0x00b2 = 178
38. 5c -> PRIMARY\_DRAWING\_ORDER::bounds::right (delta) = 0x5c = 92 pixels from the last bounds::right
39. PRIMARY\_DRAWING\_ORDER::bounds::right = last bounds::right (0x2f3) + 0x5c = 0x34f = 847
40. PRIMARY\_DRAWING\_ORDER::bounds::bottom not present
41. 36 02 -> POLYGON\_SC\_ORDER::xStart = 0x0236 = 566
42. 0d 03 -> POLYGON\_SC\_ORDER::yStart = 0x030d = 781
43. 0d -> POLYGON\_SC\_ORDER::bRop2 = 0x0d = R2\_COPYPEN
44. 01 -> POLYGON\_SC\_ORDER::FillMode = 0x01 = ALTERNATE
45. 20 ba 00 -> POLYGON\_SC\_ORDER::BrushColor
46. TS\_COLOR::RedOrPaletteIndex = 0x20
47. TS\_COLOR::Green = 0xba
48. TS\_COLOR::Blue = 0x00
49. 36 -> POLYGON\_SC\_ORDER::NumDeltaEntries = 0x36 = 54 entries
50. d0 -> VARIABLE1\_FIELD::cbData = 0xd0 = 208 bytes
51. 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 c0
52. 76 80 b5 4a 80 9f ff a1 80 81 ff 7d 80 5d ff 5f
53. 33 ff 4c 08 ff 3f 5c ff 3e ff b1 ff 46 ff 8a ff
54. 59 ff 6b ff 73 ff 52 ff 96 ff 42 ff be ff 3e 69
55. ff 41 15 ff 51 80 40 ff 68 80 69 ff 87 80 8b ff
56. ad 80 a7 58 80 b9 03 80 c2 2f 80 c1 80 59 80 b5
57. 80 7e 80 a1 80 9d 80 85 80 b1 80 60 80 c0 38 80
58. c3 0c 80 bb 60 80 ab ff b5 80 91 ff 8e 80 70 ff
59. 6e 80 49 ff 54 1e ff 44 71 ff 3d 46 ff 41 ff 9d
60. ff 4f ff 7a ff 65 ff 5d ff 83 ff 49 ff aa ff 3f
61. 53 ff 3e 7f ff 49 2b ff 5b 80 55 ff 76 80 7b ff
62. 9a 80 99 41 80 b1 6e 80 bf 19 80 c3 80 45 80 bc
63. 80 6c 80 ad 80 8e 80 94 80 a8 80 73 81 7d 80 6f -> VARIABLE1\_FIELD::rgbData
64. 00 00 00 00 00 00 00 00 00 00 00 00 00 00 -> DELTA\_PTS\_FIELD::zeroBits = 0
65. Point 1: 80 c0 -> Delta X = 0xc0 = 192 76 -> Delta Y = 0xfffffff6 = -10
66. Point 2: 80 b5 -> Delta X = 0xb5 = 181 4a -> Delta Y = 0xffffffca = -54
67. Point 3: 80 9f -> Delta X = 0x9f = 159 ff a1 -> Delta Y = 0xffffffa1 = -95
68. Point 4: 80 81 -> Delta X = 0x81 = 129 ff 7d -> Delta Y = 0xffffff7d = -131
69. Point 5: 80 5d -> Delta X = 0x5d = 93 ff 5f -> Delta Y = 0xffffff5f = -161
70. Point 6: 33 -> Delta X = 0x33 = 51 ff 4c -> Delta Y = 0xffffff4c = -180
71. Point 7: 08 -> Delta X = 0x8 = 8 ff 3f -> Delta Y = 0xffffff3f = -193
72. Point 8: 5c -> Delta X = 0xffffffdc = -36 ff 3e -> Delta Y = 0xffffff3e = -194
73. Point 9: ff b1 -> Delta X = 0xffffffb1 = -79 ff 46 -> Delta Y = 0xffffff46 = -186
74. Point 10: ff 8a -> Delta X = 0xffffff8a = -118 ff 59 -> Delta Y = 0xffffff59 = -167
75. Point 11: ff 6b -> Delta X = 0xffffff6b = -149 ff 73 -> Delta Y = 0xffffff73 = -141
76. Point 12: ff 52 -> Delta X = 0xffffff52 = -174 ff 96 -> Delta Y = 0xffffff96 = -106
77. Point 13: ff 42 -> Delta X = 0xffffff42 = -190 ff be -> Delta Y = 0xffffffbe = -66
78. Point 14: ff 3e -> Delta X = 0xffffff3e = -194 69 -> Delta Y = 0xffffffe9 = -23
79. Point 15: ff 41 -> Delta X = 0xffffff41 = -191 15 -> Delta Y = 0x15 = 21
80. Point 16: ff 51 -> Delta X = 0xffffff51 = -175 80 40 -> Delta Y = 0x40 = 64
81. Point 17: ff 68 -> Delta X = 0xffffff68 = -152 80 69 -> Delta Y = 0x69 = 105
82. Point 18: ff 87 -> Delta X = 0xffffff87 = -121 80 8b -> Delta Y = 0x8b = 139
83. Point 19: ff ad -> Delta X = 0xffffffad = -83 80 a7 -> Delta Y = 0xa7 = 167
84. Point 20: 58 -> Delta X = 0xffffffd8 = -40 80 b9 -> Delta Y = 0xb9 = 185
85. Point 21: 03 -> Delta X = 0x3 = 3 80 c2 -> Delta Y = 0xc2 = 194
86. Point 22: 2f -> Delta X = 0x2f = 47 80 c1 -> Delta Y = 0xc1 = 193
87. Point 23: 80 59 -> Delta X = 0x59 = 89 80 b5 -> Delta Y = 0xb5 = 181
88. Point 24: 80 7e -> Delta X = 0x7e = 126 80 a1 -> Delta Y = 0xa1 = 161
89. Point 25: 80 9d -> Delta X = 0x9d = 157 80 85 -> Delta Y = 0x85 = 133
90. Point 26: 80 b1 -> Delta X = 0xb1 = 177 80 60 -> Delta Y = 0x60 = 96
91. Point 27: 80 c0 -> Delta X = 0xc0 = 192 38 -> Delta Y = 0x38 = 56
92. Point 28: 80 c3 -> Delta X = 0xc3 = 195 0c -> Delta Y = 0xc = 12
93. Point 29: 80 bb -> Delta X = 0xbb = 187 60 -> Delta Y = 0xffffffe0 = -32
94. Point 30: 80 ab -> Delta X = 0xab = 171 ff b5 -> Delta Y = 0xffffffb5 = -75
95. Point 31: 80 91 -> Delta X = 0x91 = 145 ff 8e -> Delta Y = 0xffffff8e = -114
96. Point 32: 80 70 -> Delta X = 0x70 = 112 ff 6e -> Delta Y = 0xffffff6e = -146
97. Point 33: 80 49 -> Delta X = 0x49 = 73 ff 54 -> Delta Y = 0xffffff54 = -172
98. Point 34: 1e -> Delta X = 0x1e = 30 ff 44 -> Delta Y = 0xffffff44 = -188
99. Point 35: 71 -> Delta X = 0xfffffff1 = -15 ff 3d -> Delta Y = 0xffffff3d = -195
100. Point 36: 46 -> Delta X = 0xffffffc6 = -58 ff 41 -> Delta Y = 0xffffff41 = -191
101. Point 37: ff 9d -> Delta X = 0xffffff9d = -99 ff 4f -> Delta Y = 0xffffff4f = -177
102. Point 38: ff 7a -> Delta X = 0xffffff7a = -134 ff 65 -> Delta Y = 0xffffff65 = -155
103. Point 39: ff 5d -> Delta X = 0xffffff5d = -163 ff 83 -> Delta Y = 0xffffff83 = -125
104. Point 40: ff 49 -> Delta X = 0xffffff49 = -183 ff aa -> Delta Y = 0xffffffaa = -86
105. Point 41: ff 3f -> Delta X = 0xffffff3f = -193 53 -> Delta Y = 0xffffffd3 = -45
106. Point 42: ff 3e -> Delta X = 0xffffff3e = -194 7f -> Delta Y = 0xffffffff = -1
107. Point 43: ff 49 -> Delta X = 0xffffff49 = -183 2b -> Delta Y = 0x2b = 43
108. Point 44: ff 5b -> Delta X = 0xffffff5b = -165 80 55 -> Delta Y = 0x55 = 85
109. Point 45: ff 76 -> Delta X = 0xffffff76 = -138 80 7b -> Delta Y = 0x7b = 123
110. Point 46: ff 9a -> Delta X = 0xffffff9a = -102 80 99 -> Delta Y = 0x99 = 153
111. Point 47: 41 -> Delta X = 0xffffffc1 = -63 80 b1 -> Delta Y = 0xb1 = 177
112. Point 48: 6e -> Delta X = 0xffffffee = -18 80 bf -> Delta Y = 0xbf = 191
113. Point 49: 19 -> Delta X = 0x19 = 25 80 c3 -> Delta Y = 0xc3 = 195
114. Point 50: 80 45 -> Delta X = 0x45 = 69 80 bc -> Delta Y = 0xbc = 188
115. Point 51: 80 6c -> Delta X = 0x6c = 108 80 ad -> Delta Y = 0xad = 173
116. Point 52: 80 8e -> Delta X = 0x8e = 142 80 94 -> Delta Y = 0x94 = 148
117. Point 53: 80 a8 -> Delta X = 0xa8 = 168 80 73 -> Delta Y = 0x73 = 115
118. Point 54: 81 7d -> Delta X = 0x17d = 381 80 6f -> Delta Y = 0x6f = 111
119. Actual point 1: 758, 771
120. Actual point 2: 939, 717
121. Actual point 3: 1098, 622
122. Actual point 4: 1227, 491
123. Actual point 5: 1320, 330
124. Actual point 6: 1371, 150
125. Actual point 7: 1379, -43
126. Actual point 8: 1343, -237
127. Actual point 9: 1264, -423
128. Actual point 10: 1146, -590
129. Actual point 11: 997, -731
130. Actual point 12: 823, -837
131. Actual point 13: 633, -903
132. Actual point 14: 439, -926
133. Actual point 15: 248, -905
134. Actual point 16: 73, -841
135. Actual point 17: -79, -736
136. Actual point 18: -200, -597
137. Actual point 19: -283, -430
138. Actual point 20: -323, -245
139. Actual point 21: -320, -51
140. Actual point 22: -273, 142
141. Actual point 23: -184, 323
142. Actual point 24: -58, 484
143. Actual point 25: 99, 617
144. Actual point 26: 276, 713
145. Actual point 27: 468, 769
146. Actual point 28: 663, 781
147. Actual point 29: 850, 749
148. Actual point 30: 1021, 674
149. Actual point 31: 1166, 560
150. Actual point 32: 1278, 414
151. Actual point 33: 1351, 242
152. Actual point 34: 1381, 54
153. Actual point 35: 1366, -141
154. Actual point 36: 1308, -332
155. Actual point 37: 1209, -509
156. Actual point 38: 1075, -664
157. Actual point 39: 912, -789
158. Actual point 40: 729, -875
159. Actual point 41: 536, -920
160. Actual point 42: 342, -921
161. Actual point 43: 159, -878
162. Actual point 44: -6, -793
163. Actual point 45: -144, -670
164. Actual point 46: -246, -517
165. Actual point 47: -309, -340
166. Actual point 48: -327, -149
167. Actual point 49: -302, 46
168. Actual point 50: -233, 234
169. Actual point 51: -125, 407
170. Actual point 52: 17, 555
171. Actual point 53: 185, 670
172. Actual point 54: 566, 781

### PolygonCB

The following is an annotated dump of a [PolygonCB (section 2.2.2.2.1.1.2.17)](#Section_24f4e2291f9f4633b9be66d7579ba36d) Primary Drawing Order.

1. 00000000 09 15 ef 1b ea 00 46 01 0d 01 08 00 00 04 03 81 ......F.........
2. 00000010 08 03 05 88 09 26 09 77 .....&.w
3. 09 -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
4. 0x09
5. = 0x01 |
6. 0x08
7. = TS\_STANDARD |
8. TS\_TYPE\_CHANGE
9. 15 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x15 = TS\_ENC\_POLYGON\_CB\_ORDER
10. ef 1b -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x1bef
11. Binary of 0x1bef = 0001 1011 1110 1111
12. Fields 1-4, 6-10, 12-13 are present
13. ea 00 -> POLYGON\_CB\_ORDER::xStart = 0xea = 234
14. 46 01 -> POLYGON\_CB\_ORDER::yStart = 0x146 = 326
15. 0d -> POLYGON\_CB\_ORDER::bRop2 = 0x0d = COPYPEN
16. 01 -> POLYGON\_CB\_ORDER::FillMode = 0x01 = ALTERNATE
17. POLYGON\_CB\_ORDER::BackColor not present
18. 08 00 00 -> POLYGON\_CB\_ORDER::ForeColor
19. TS\_COLOR::RedOrPaletteIndex = 0x08
20. TS\_COLOR::Green = 0x00
21. TS\_COLOR::Blue = 0x00
22. 04 -> POLYGON\_CB\_ORDER::BrushOrgX = 4
23. 03 -> POLYGON\_CB\_ORDER::BrushOrgY = 3
24. 81 -> POLYGON\_CB\_ORDER:BrushStyle = 0x81 = 0x80 | 0x01 = TS\_CACHED\_BRUSH | BMF\_1BPP
25. 08 -> POLYGON\_CB\_ORDER:BrushHatch = Brush Cache Index = 0x8
26. POLYGON\_CB\_ORDER::BrushExtra not present
27. 03 -> POLYGON\_CB\_ORDER::NumDeltaEntries = 0x3 = 3 entries
28. 05 -> VARIABLE1\_FIELD::cbData = 5 bytes
29. 88 09 26 09 77 -> VARIABLE1\_FIELD::rgbData
30. 88 -> DELTA\_PTS\_FIELD::zeroBits = binary:10 00 10 00
31. Point 1: Delta X = 0 (zeroBit set) 09 -> Delta Y = 0x9 = 9
32. Point 2: 26 -> Delta X = 0x26 = 38 09 -> Delta Y = 0x9 = 9
33. Point 3: Delta X = 0 (zeroBit set) 77 -> Delta Y = 0xfffffff7 = -9
34. Actual point 1: 234, 335
35. Actual point 2: 272, 344
36. Actual point 3: 272, 335

### Polyline

The following is an annotated dump of a [Polyline (section 2.2.2.2.1.1.2.18)](#Section_b60b4d9baac54b9884d3edc0c0779a5c) Primary Drawing Order.

1. 00000000 2d 16 73 f8 01 b8 02 00 c0 00 20 6c 00 00 00 00 -.s....... l....
2. 00000010 00 04 00 00 ff 7e 76 ff 41 6c ff 24 62 ff 2b 59 .....~v.Al.$b.+Y
3. 00000020 ff 55 51 ff 9c 49 73 43 80 4d ff be 80 99 ff ba .UQ..IsC.M......
4. 00000030 80 cd ff b7 80 de ff b6 80 ca ff b6 80 96 ff b7 ................
5. 00000040 80 48 ff ba 6f ff be ff 97 43 ff 52 4a ff 2b 51 .H..o....C.RJ.+Q
6. 00000050 ff 24 59 ff 44 63 ff 81 6c 56 76 2f 80 82 0a 80 .$Y.Dc..lVv/....
7. 00000060 bf 14 80 dd 1e 80 d4 27 80 ab 2f 80 64 37 0d 3d .......'../.d7.=
8. 00000070 ff b3 80 42 ff 67 80 46 ...B.g.F
9. 2d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x2d
10. 0x2d
11. = 0x01 |
12. 0X04 |
13. 0x08 |
14. 0x20
15. = TS\_STANDARD |
16. TS\_BOUNDS |
17. TS\_TYPE\_CHANGE |
18. TS\_ZERO\_BOUNDS\_DELTAS
19. 16 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x16 = TS\_ENC\_POLYLINE\_ORDER
20. 73 -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x73
21. Binary of 0x73 = 1110011
22. Fields 1-2, 5-7 are present
23. PRIMARY\_DRAWING\_ORDER::bounds = last bounds
24. f8 01 -> POLYLINE\_ORDER::xStart = 0x01f8 = 504
25. b8 02 -> POLYLINE\_ORDER::yStart = 0x02b8 = 696
26. POLYLINE\_ORDER::bRop2 not present
27. POLYLINE\_ORDER::BrushCacheEntry not present
28. 00 c0 00 -> POLYLINE\_ORDER::PenColor::Red = 0x00
29. TS\_COLOR::RedOrPaletteIndex = 0x00
30. TS\_COLOR::Green = 0xc0
31. TS\_COLOR::Blue = 0x00
32. 20 -> POLYLINE\_ORDER::NumDeltaEntries = 0x20 = 32 entries
33. 6c -> VARIABLE1\_FIELD::cbData = 0x6c = 108 bytes
34. 00 00 00 00 00 04 00 00 ff 7e 76 ff 41 6c ff 24
35. 62 ff 2b 59 ff 55 51 ff 9c 49 73 43 80 4d ff be
36. 80 99 ff ba 80 cd ff b7 80 de ff b6 80 ca ff b6
37. 80 96 ff b7 80 48 ff ba 6f ff be ff 97 43 ff 52
38. 4a ff 2b 51 ff 24 59 ff 44 63 ff 81 6c 56 76 2f
39. 80 82 0a 80 bf 14 80 dd 1e 80 d4 27 80 ab 2f 80
40. 64 37 0d 3d ff b3 80 42 ff 67 80 46 -> VARIABLE1\_FIELD::rgbData
41. 00 00 00 00 00 04 00 00 -> DELTA\_PTS\_FIELD::zeroBits
42. Point 1: ff 7e -> Delta X = 0xffffff7e = -130 76 -> Delta Y = 0xfffffff6 = -10
43. Point 2: ff 41 -> Delta X = 0xffffff41 = -191 6c -> Delta Y = 0xffffffec = -20
44. Point 3: ff 24 -> Delta X = 0xffffff24 = -220 62 -> Delta Y = 0xffffffe2 = -30
45. Point 4: ff 2b -> Delta X = 0xffffff2b = -213 59 -> Delta Y = 0xffffffd9 = -39
46. Point 5: ff 55 -> Delta X = 0xffffff55 = -171 51 -> Delta Y = 0xffffffd1 = -47
47. Point 6: ff 9c -> Delta X = 0xffffff9c = -100 49 -> Delta Y = 0xffffffc9 = -55
48. Point 7: 73 -> Delta X = 0xfffffff3 = -13 43 -> Delta Y = 0xffffffc3 = -61
49. Point 8: 80 4d -> Delta X = 0x4d = 77 ff be -> Delta Y = 0xffffffbe = -66
50. Point 9: 80 99 -> Delta X = 0x99 = 153 ff ba -> Delta Y = 0xffffffba = -70
51. Point 10: 80 cd -> Delta X = 0xcd = 205 ff b7 -> Delta Y = 0xffffffb7 = -73
52. Point 11: 80 de -> Delta X = 0xde = 222 ff b6 -> Delta Y = 0xffffffb6 = -74
53. Point 12: 80 ca -> Delta X = 0xca = 202 ff b6 -> Delta Y = 0xffffffb6 = -74
54. Point 13: 80 96 -> Delta X = 0x96 = 150 ff b7 -> Delta Y = 0xffffffb7 = -73
55. Point 14: 80 48 -> Delta X = 0x48 = 72 ff ba -> Delta Y = 0xffffffba = -70
56. Point 15: 6f -> Delta X = 0xffffffef = -17 ff be -> Delta Y = 0xffffffbe = -66
57. Point 16: ff 97 -> Delta X = 0xffffff97 = -105 43 -> Delta Y = 0xffffffc3 = -61
58. Point 17: ff 52 -> Delta X = 0xffffff52 = -174 4a -> Delta Y = 0xffffffca = -54
59. Point 18: ff 2b -> Delta X = 0xffffff2b = -213 51 -> Delta Y = 0xffffffd1 = -47
60. Point 19: ff 24 -> Delta X = 0xffffff24 = -220 59 -> Delta Y = 0xffffffd9 = -39
61. Point 20: ff 44 -> Delta X = 0xffffff44 = -188 63 -> Delta Y = 0xffffffe3 = -29
62. Point 21: ff 81 -> Delta X = 0xffffff81 = -127 6c -> Delta Y = 0xffffffec = -20
63. Point 22: 56 -> Delta X = 0xffffffd6 = -42 76 -> Delta Y = 0xfffffff6 = -10
64. Point 23: 2f -> Delta X = 0x2f = 47 Delta Y = 0 (due to zeroBit being set)
65. Point 24: 80 82 -> Delta X = 0x82 = 130 0a -> Delta Y = 0xa = 10
66. Point 25: 80 bf -> Delta X = 0xbf = 191 14 -> Delta Y = 0x14 = 20
67. Point 26: 80 dd -> Delta X = 0xdd = 221 1e -> Delta Y = 0x1e = 30
68. Point 27: 80 d4 -> Delta X = 0xd4 = 212 27 -> Delta Y = 0x27 = 39
69. Point 28: 80 ab -> Delta X = 0xab = 171 2f -> Delta Y = 0x2f = 47
70. Point 29: 80 64 -> Delta X = 0x64 = 100 37 -> Delta Y = 0x37 = 55
71. Point 30: 0d -> Delta X = 0xd = 13 3d -> Delta Y = 0x3d = 61
72. Point 31: ff b3 -> Delta X = 0xffffffb3 = -77 80 42 -> Delta Y = 0x42 = 66
73. Point 32: ff 67 -> Delta X = 0xffffff67 = -153 80 46 -> Delta Y = 0x46 = 70
74. Actual point 1: 374, 686
75. Actual point 2: 183, 666
76. Actual point 3: -37, 636
77. Actual point 4: -250, 597
78. Actual point 5: -421, 550
79. Actual point 6: -521, 495
80. Actual point 7: -534, 434
81. Actual point 8: -457, 368
82. Actual point 9: -304, 298
83. Actual point 10: -99, 225
84. Actual point 11: 123, 151
85. Actual point 12: 325, 77
86. Actual point 13: 475, 4
87. Actual point 14: 547, -66
88. Actual point 15: 530, -132
89. Actual point 16: 425, -193
90. Actual point 17: 251, -247
91. Actual point 18: 38, -294
92. Actual point 19: -182, -333
93. Actual point 20: -370, -362
94. Actual point 21: -497, -382
95. Actual point 22: -539, -392
96. Actual point 23: -492, -392
97. Actual point 24: -362, -382
98. Actual point 25: -171, -362
99. Actual point 26: 50, -332
100. Actual point 27: 262, -293
101. Actual point 28: 433, -246
102. Actual point 29: 533, -191
103. Actual point 30: 546, -130
104. Actual point 31: 469, -64
105. Actual point 32: 316, 6

### EllipseSC

The following is an annotated dump of an [EllipseSC (section 2.2.2.2.1.1.2.19)](#Section_5146fab3c43f44d7b54104666a80e33b) Primary Drawing Order.

1. 00000000 0d 19 7f 29 ca 00 fb 3c 02 ca 00 47 01 eb 03 3c ...)...<...G...<
2. 00000010 02 0d 02 c0 21 00 ....!.
3. 0d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
4. 0x0d
5. = 0x01 |
6. 0x04 |
7. 0x08
8. = TS\_STANDARD |
9. TS\_BOUNDS |
10. TS\_TYPE\_CHANGE
11. 19 -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_ELLIPSE\_SC\_ORDER
12. 7f -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x7f
13. Binary of 0x7f = 0111 1111
14. Fields 1-7 are present
15. 29 -> PRIMARY\_DRAWING\_ORDER::bounds::flags = 0x29
16. 0x29
17. = 0x01 |
18. 0x08 |
19. 0x20
20. = TS\_BOUND\_LEFT |
21. TS\_BOUND\_BOTTOM |
22. TS\_BOUND\_DELTA\_TOP
23. ca 00 -> PRIMARY\_DRAWING\_ORDER::bounds::left = 0xca = 202
24. fb -> PRIMARY\_DRAWING\_ORDER::bounds::top(delta) = 0xfb = -5 pixels from last bounds::top
25. PRIMARY\_DRAWING\_ORDER::bounds::top = last bounds::top (0x14c) + 0xfb = 0x147 = 327
26. PRIMARY\_DRAWING\_ORDER::bounds::right not present
27. 3c 02 -> PRIMARY\_DRAWING\_ORDER::bounds::bottom = 0x023c = 572
28. ca 00 -> ELLIPSE\_SC\_ORDER::LeftRect = 0x00ca = 202
29. 47 01 -> ELLIPSE\_SC\_ORDER::TopRect = 0x0147 = 327
30. eb 03 -> ELLIPSE\_SC\_ORDER::RightRect = 0x03eb = 1003
31. 3c 02 -> ELLIPSE\_SC\_ORDER::BottomRect = 0x023c = 572
32. 0d -> ELLIPSE\_SC\_ORDER::bRop2 = 0x0d = COPYPEN
33. 02 -> ELLIPSE\_SC\_ORDER::FillMode = 0x02 = WINDING
34. c0 21 00 -> ELLIPSE\_SC\_ORDER::Color
35. TS\_COLOR::RedOrPaletteIndex = 0xc0
36. TS\_COLOR::Green = 0x21
37. TS\_COLOR::Blue = 0x00

### EllipseCB

The following is an annotated dump of an [EllipseCB (section 2.2.2.2.1.1.2.20)](#Section_897203838394423a810aba631d66055f) Primary Drawing Order.

1. 00000000 0d 1a ff 0c b0 03 b9 c9 09 00 8e fe 10 01 5e 02 ..............^.
2. 00000010 0d 01 ff ff 00 60 30 00 02 05 .....`0...
3. 0d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x09
4. 0x0d
5. = 0x01 |
6. 0x04 |
7. 0x08
8. = TS\_STANDARD |
9. TS\_BOUNDS |
10. TS\_TYPE\_CHANGE
11. 1a -> PRIMARY\_DRAWING\_ORDER::orderType = TS\_ENC\_ELLIPSE\_CB\_ORDER
12. ff 0c -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x0cff
13. Binary of 0x0cff = 0000 1100 1111 1111
14. Fields 1-8, 11-12 are present
15. b0 -> PRIMARY\_DRAWING\_ORDER::bounds::flags = 0x29
16. 0xb0
17. = 0x10 |
18. 0x20 |
19. 0x80
20. = TS\_BOUND\_DELTA\_LEFT |
21. TS\_BOUND\_DELTA\_TOP |
22. TS\_BOUND\_DELTA\_BOTTOM
23. 03 -> PRIMARY\_DRAWING\_ORDER::bounds::left(delta) = 0x03 = 3 pixels from last bounds::left
24. PRIMARY\_DRAWING\_ORDER::bounds::left = last bounds::left (0x2e) + 0x03 = 0x32 = 49
25. b9 -> PRIMARY\_DRAWING\_ORDER::bounds::top(delta) = 0xb9 = -71 pixels from last bounds::top
26. PRIMARY\_DRAWING\_ORDER::bounds::top = last bounds::top (0x89) + 0xb9 = 0x42 = 66
27. PRIMARY\_DRAWING\_ORDER::bounds::right not present
28. c9 -> PRIMARY\_DRAWING\_ORDER::bounds::bottom(delta) = 0xc9 = -55 pixels from last bounds::bottom
29. PRIMARY\_DRAWING\_ORDER::bounds::bottom = last bounds::bottom (0xad) + 0xc9 = 0x76 = 118
30. 09 00 -> ELLIPSE\_CB\_ORDER::LeftRect = 0x0009 = 9
31. 8e fe -> ELLIPSE\_CB\_ORDER::TopRect = 0xfe8e = -370
32. 10 01 -> ELLIPSE\_CB\_ORDER::RighttRect = 0x0110 = 272
33. 5e 02 -> ELLIPSE\_CB\_ORDER::BottomRect = 0x025e = 606
34. 0d -> ELLIPSE\_CB\_ORDER::bRop2 = 0x0d = COPYPEN
35. 01 -> ELLIPSE\_CB\_ORDER::FillMode = 0x01 = ALTERNATE
36. ff ff 00 -> ELLIPSE\_CB\_ORDER::BackColor
37. TS\_COLOR::RedOrPaletteIndex = 0xff
38. TS\_COLOR::Green = 0xff
39. TS\_COLOR::Blue = 0x00
40. 60 30 00 -> ELLIPSE\_CB\_ORDER::ForeColor::Red = 0xff
41. TS\_COLOR::RedOrPaletteIndex = 0x60
42. TS\_COLOR::Green = 0x30
43. TS\_COLOR::Blue = 0x00
44. ELLIPSE\_CB\_ORDER::BrushOrgX not present
45. ELLIPSE\_CB\_ORDER::BrushOrgY not present
46. 02 -> ELLIPSE\_CB\_ORDER::BrushStyle = 0x02 = BMF\_4BPP
47. 05 -> ELLIPSE\_CB\_ORDER::BrushHatch = Brush Cache Index = 0x05
48. ELLIPSE\_CB\_ORDER::BrushExtra not present

### DrawNineGrid

The following is an annotated dump of a [DrawNineGrid (section 2.2.2.2.1.1.2.21)](#Section_44bd6a68a5b74a38980706ba4ab4a72a) Primary Drawing Order.

1. 00000000 1d 07 1c 87 39 03 86 00 4e 03 7d fb f9 0d ....9...N.}...
2. 1d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x1d
3. 0x1d
4. = 0x01 |
5. 0X04 |
6. 0x08 |
7. 0x10
8. = TS\_STANDARD |
9. TS\_BOUNDS |
10. TS\_TYPE\_CHANGE |
11. TS\_DELTA\_COORDINATES
12. 07 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x07 = TS\_ENC\_DRAWNINEGRID\_ORDER
13. 1c -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x1c
14. Binary of 0x1c = 0001 1100
15. Fields 3-5 are present
16. 87 -> PRIMARY\_DRAWING\_ORDER::bounds::flags = 0x87
17. 0x87
18. = 0x01 |
19. 0x02 |
20. 0x04 |
21. 0x80
22. = TS\_BOUND\_LEFT |
23. TS\_BOUND\_TOP |
24. TS\_BOUND\_RIGHT |
25. TS\_BOUND\_DELTA\_BOTTOM
26. 39 03 -> PRIMARY\_DRAWING\_ORDER::bounds::left = 0x0393 = 925
27. 86 00 -> PRIMARY\_DRAWING\_ORDER::bounds::top = 0x86 = 134
28. 4e 03 -> PRIMARY\_DRAWING\_ORDER::bounds::right = 0x034e = 846
29. 7d -> PRIMARY\_DRAWING\_ORDER::bounds::bottom(delta) = 0x7d = -131 pixels from the last bounds::bottom
30. PRIMARY\_DRAWING\_ORDER::bounds::bottom = last bounds::bottom (0x11e) + 0x7d = 0x9b = 155
31. DRAWNINEGRID\_ORDER\_Order::srcLeft not present
32. DRAWNINEGRID\_ORDER\_Order::srcTop not present
33. fb -> DRAWNINEGRID\_ORDER\_Order::srcRight (delta) = 0xfb = -5 pixels from last srcRight
34. DRAWNINEGRID\_ORDER\_Order::srcRight = last DRAWNINEGRID\_ORDER\_Order::srcRight (0x26) + 0xfb = 0x21
35. f9 -> DRAWNINEGRID\_ORDER\_Order::srcBottom (delta) = 0xf9 = -7 pixels from last srcBottom
36. DRAWNINEGRID\_ORDER\_Order::srcBottom = last DRAWNINEGRID\_ORDER\_Order::srcBottom (0x28) + 0xf9 = 0x21
37. 0d -> DRAWNINEGRID\_ORDER\_Order::bitmapId = 13

### MultiDrawNineGrid

The following is an annotated dump of a [MultiDrawNineGrid (section 2.2.2.2.1.1.2.22)](#Section_101fecdc3ad841c6b7b670d8efd4ad70) Primary Drawing Order.

1. 00000000 1d 08 7c 03 9b 00 7c fe 03 02 a4 00 01 05 00 40 ..|...|........@
2. 00000010 80 9b 05 1a ....
3. 1d -> PRIMARY\_DRAWING\_ORDER::controlFlags = 0x1d
4. 0x1d
5. = 0x01 |
6. 0X04 |
7. 0x08 |
8. 0x10
9. = TS\_STANDARD |
10. TS\_BOUNDS |
11. TS\_TYPE\_CHANGE |
12. TS\_DELTA\_COORDINATES
13. 08 -> PRIMARY\_DRAWING\_ORDER::orderType = 0x08 = TS\_ENC\_MULTI\_DRAWNINEGRID\_ORDER
14. 7c -> PRIMARY\_DRAWING\_ORDER::fieldFlags = 0x7c
15. Binary of 0x7c = 1111100
16. Fields 3-7 are present
17. 03 -> PRIMARY\_DRAWING\_ORDER::bounds::flags = 0x3
18. 0x3
19. = 0x1 |
20. 0x2
21. = TS\_BOUND\_LEFT |
22. TS\_BOUND\_TOP
23. 9b 00 -> PRIMARY\_DRAWING\_ORDER::bounds::left = 0x9b = 155
24. 7c fe -> PRIMARY\_DRAWING\_ORDER::bounds::top = 0xfec7 = -388
25. PRIMARY\_DRAWING\_ORDER::bounds::right not present
26. PRIMARY\_DRAWING\_ORDER::bounds::bottom not present
27. MULTI\_DRAWNINEGRID\_ORDER\_Order::srcLeft not present
28. MULTI\_DRAWNINEGRID\_ORDER\_Order::srcTop not present
29. 03 -> MULTI\_DRAWNINEGRID\_ORDER\_Order::srcRight (delta) = 0x3 = 3 pixels from last srcRight
30. MULTI\_DRAWNINEGRID\_ORDER\_Order::srcRight = last MULTI\_DRAWNINEGRID\_ORDER\_Order::srcRight (0x02) + 0x3 = 5
31. 02 -> MULTI\_DRAWNINEGRID\_ORDER\_Order::srcBottom (delta) = 0x2 = 2 pixels from last srcBottom
32. MULTI\_DRAWNINEGRID\_ORDER\_Order::srcBottom = last MULTI\_DRAWNINEGRID\_ORDER\_Order::srcBottom (0x01) + 0x2 = 3
33. a4 00 -> MULTI\_DRAWNINEGRID\_ORDER\_Order::bitmapId = 0xa4
34. 01 -> MULTI\_DRAWNINEGRID\_ORDER\_Order::nDeltaEntries = 0x01 = 1 entry
35. 05 00 -> VARIABLE2\_FIELD::cbData = 0x0005 = 5 bytes
36. 40 80 9b 05 1a-> VARIABLE2\_FIELD::rgbData
37. 40 -> DELTA\_RECTS\_FIELD::zeroBits = binary:0100 0000
38. Rectangle #1:
39. 80 9b -> Delta Left = 0x9b = 155
40. Delta Top = 0 (zeroBit is set)
41. 05 -> Delta Width = 0x5 = 5
42. 1a -> Delta Height = 0x1a = 26
43. Rectangle is (155, 0, 155 + 5 = 160, 0 + 26 = 26)

## Annotated Secondary Drawing Orders

### Cache Bitmap (Revision 1)

The following is an annotated dump of a Cache Bitmap - Revision 1 (section [2.2.2.2.1.2.2)](#Section_c9365b320be547c9af1afeba2ea1ee9f) Secondary Drawing Order.

1. 00000000 03 03 00 00 04 02 00 00 10 01 08 01 00 00 00 10 ................
2. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
3. 0x03
4. = 0x2 |
5. 0x1
6. = TS\_SECONDARY |
7. TS\_STANDARD
8. 03 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0x0003 + 13 = 16 bytes
9. 00 04 -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = 0x0400 = NO\_BITMAP\_COMPRESSION\_HDR
10. 02 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = 0x02 = TS\_CACHE\_BITMAP\_COMPRESSED
11. 00 -> CACHE\_BITMAP\_ORDER::cacheId = 0
12. 00 -> CACHE\_BITMAP\_ORDER::pad1Octet
13. 10 -> CACHE\_BITMAP\_ORDER::bitmapWidth = 0x10 = 16
14. 01 -> CACHE\_BITMAP\_ORDER::bitmapHeight = 0x01 = 1
15. 08 -> CACHE\_BITMAP\_ORDER::bitmapBitsPerPel = 0x08 = 8 bpp
16. 01 00 -> CACHE\_BITMAP\_ORDER::bitmapLength = 0x0001 = 1 byte
17. 00 00 -> CACHE\_BITMAP\_ORDER::cacheIndex = 0
18. Compressed bitmap data (1 byte)
19. 00000000 10 .
20. Decompressed bitmap data:
21. 00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

### Cache Bitmap (Revision 2)

The following is an annotated dump of a Cache Bitmap - Revision 2 (section [2.2.2.2.1.2.3)](#Section_3c76f8fbfcaa4ca69d2e3f449bdbed40) Secondary Drawing Order.

1. 00000000 03 da 00 a1 0c 05 20 40 dc ff ff 85 ff ff 99 d6 ...... @........
2. 00000010 99 d6 99 d6 99 d6 06 8b 99 d6 99 d6 99 d6 10 84 ................
3. 00000020 08 42 08 42 10 84 99 d6 99 d6 99 d6 99 d6 06 84 .B.B............
4. 00000030 99 d6 99 d6 99 d6 ff ff 16 69 99 d6 06 69 99 d6 .........i...i..
5. 00000040 04 cc 89 52 03 6e ff ff 02 6e 08 42 01 70 08 42 ...R.n...n.B.p.B
6. 00000050 71 ff ff ce 18 c6 01 81 08 42 ce 66 29 02 cd 89 q........B.f)...
7. 00000060 52 03 88 10 84 99 d6 99 d6 99 d6 00 00 00 00 00 R...............
8. 00000070 00 00 00 d8 99 d6 03 f8 01 00 00 00 00 f0 66 99 ..............f.
9. 00000080 d6 05 6a 99 d6 00 c4 cc 89 52 03 6e ff ff 02 6e ..j......R.n...n
10. 00000090 08 42 01 70 08 42 71 ff ff ce 18 c6 01 81 08 42 .B.p.Bq........B
11. 000000a0 ce 66 29 02 cd 89 52 03 00 04 d6 99 d6 c3 80 61 .f)...R........a
12. 000000b0 00 a5 80 40 ec 52 00 5a 00 2d 00 24 00 12 00 24 ...@.R.Z.-.$...$
13. 000000c0 00 12 00 5a 00 2d 00 a5 80 52 00 c3 80 61 00 00 ...Z.-...R...a..
14. 000000d0 00 00 00 cc 89 52 03 6e ff ff 02 cb 18 c6 84 08 .....R.n........
15. 000000e0 42 08 42 08 42 ff ff B.B.B..
16. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
17. 0x03
18. = 0x2 |
19. 0x1
20. = TS\_SECONDARY |
21. TS\_STANDARD
22. da 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0xda + 13 = 231 bytes
23. a1 0c -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = 0x0ca1 = binary:000011001 0100 001
24. cacheId = binary:001 = 1
25. bitsPerPixelID = binary:0100 = 0x4 (16 bpp)
26. flags = binary:11001 = 0x19
27. 0x19
28. = 0x10 |
29. 0x08 |
30. 0x01
31. = CBR2\_DO\_NOT\_CACHE |
32. CBR2\_NO\_BITMAP\_COMPRESSION\_HEADER |
33. CBR2\_HEIGHT\_SAME\_AS\_WIDTH
34. 05 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = 0x05 = TS\_CACHE\_BITMAP\_COMPRESSED\_REV2
35. 20 -> CACHE\_BITMAP\_REV2\_ORDER::bitmapWidth = 0x20 = 32
36. CACHE\_BITMAP\_REV2\_ORDER::bitmapHeight = 32 (CBR2\_HEIGHT\_SAME\_AS\_WIDTH flag is set)
37. 40 dc -> CACHE\_BITMAP\_REV2\_ORDER::bitmapLength = 0xdc = 220 bytes
38. ff ff -> CACHE\_BITMAP\_REV2\_ORDER::cacheIndex = 0x7fff = 32767
39. Compressed bitmap data (220 bytes):
40. 00000000 85 ff ff 99 d6 99 d6 99 d6 99 d6 06 8b 99 d6 99 ................
41. 00000010 d6 99 d6 10 84 08 42 08 42 10 84 99 d6 99 d6 99 ......B.B.......
42. 00000020 d6 99 d6 06 84 99 d6 99 d6 99 d6 ff ff 16 69 99 ..............i.
43. 00000030 d6 06 69 99 d6 04 cc 89 52 03 6e ff ff 02 6e 08 ..i.....R.n...n.
44. 00000040 42 01 70 08 42 71 ff ff ce 18 c6 01 81 08 42 ce B.p.Bq........B.
45. 00000050 66 29 02 cd 89 52 03 88 10 84 99 d6 99 d6 99 d6 f)...R..........
46. 00000060 00 00 00 00 00 00 00 00 d8 99 d6 03 f8 01 00 00 ................
47. 00000070 00 00 f0 66 99 d6 05 6a 99 d6 00 c4 cc 89 52 03 ...f...j......R.
48. 00000080 6e ff ff 02 6e 08 42 01 70 08 42 71 ff ff ce 18 n...n.B.p.Bq....
49. 00000090 c6 01 81 08 42 ce 66 29 02 cd 89 52 03 00 04 d6 ....B.f)...R....
50. 000000a0 99 d6 c3 80 61 00 a5 80 40 ec 52 00 5a 00 2d 00 ....a...@.R.Z.-.
51. 000000b0 24 00 12 00 24 00 12 00 5a 00 2d 00 a5 80 52 00 $...$...Z.-...R.
52. 000000c0 c3 80 61 00 00 00 00 00 cc 89 52 03 6e ff ff 02 ..a.......R.n...
53. 000000d0 cb 18 c6 84 08 42 08 42 08 42 ff ff .....B.B.B..
54. Decompressed bitmap data:
55. 00000000 ff ff 99 d6 99 d6 99 d6 99 d6 00 00 00 00 00 00 ................
56. 00000010 00 00 00 00 00 00 99 d6 99 d6 99 d6 10 84 08 42 ...............B
57. 00000020 08 42 10 84 99 d6 99 d6 99 d6 99 d6 00 00 00 00 .B..............
58. 00000030 00 00 00 00 00 00 00 00 99 d6 99 d6 99 d6 ff ff ................
59. 00000040 ff ff 99 d6 99 d6 99 d6 99 d6 00 00 00 00 00 00 ................
60. 00000050 00 00 00 00 00 00 99 d6 99 d6 99 d6 10 84 08 42 ...............B
61. 00000060 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
62. 00000070 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
63. 00000080 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
64. 00000090 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
65. 000000a0 08 42 10 84 10 84 10 84 10 84 10 84 10 84 10 84 .B..............
66. 000000b0 10 84 10 84 10 84 10 84 10 84 10 84 99 d6 ff ff ................
67. 000000c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
68. 000000d0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff 08 42 ...............B
69. 000000e0 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
70. 000000f0 08 42 08 42 08 42 08 42 08 42 08 42 08 42 ff ff .B.B.B.B.B.B.B..
71. 00000100 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
72. 00000110 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
73. 00000120 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
74. 00000130 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
75. 00000140 ff ff 10 84 10 84 10 84 10 84 10 84 10 84 10 84 ................
76. 00000150 10 84 10 84 10 84 10 84 10 84 10 84 10 84 08 42 ...............B
77. 00000160 08 42 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
78. 00000170 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
79. 00000180 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
80. 00000190 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
81. 000001a0 08 42 10 84 99 d6 99 d6 99 d6 00 00 00 00 00 00 .B..............
82. 000001b0 00 00 00 00 00 00 99 d6 99 d6 99 d6 99 d6 ff ff ................
83. 000001c0 ff ff 99 d6 99 d6 99 d6 00 00 00 00 00 00 00 00 ................
84. 000001d0 00 00 00 00 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
85. 000001e0 08 42 10 84 99 d6 99 d6 99 d6 00 00 00 00 00 00 .B..............
86. 000001f0 00 00 00 00 00 00 99 d6 99 d6 99 d6 99 d6 ff ff ................
87. 00000200 ff ff 99 d6 99 d6 99 d6 00 00 00 00 00 00 00 00 ................
88. 00000210 00 00 00 00 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
89. 00000220 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
90. 00000230 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
91. 00000240 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
92. 00000250 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
93. 00000260 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
94. 00000270 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
95. 00000280 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
96. 00000290 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
97. 000002a0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
98. 000002b0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
99. 000002c0 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
100. 000002d0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
101. 000002e0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
102. 000002f0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
103. 00000300 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
104. 00000310 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
105. 00000320 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
106. 00000330 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
107. 00000340 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
108. 00000350 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
109. 00000360 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
110. 00000370 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
111. 00000380 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
112. 00000390 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
113. 000003a0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
114. 000003b0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
115. 000003c0 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
116. 000003d0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
117. 000003e0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
118. 000003f0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
119. 00000400 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
120. 00000410 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
121. 00000420 08 42 10 84 10 84 10 84 10 84 10 84 10 84 10 84 .B..............
122. 00000430 10 84 10 84 10 84 10 84 10 84 10 84 99 d6 ff ff ................
123. 00000440 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
124. 00000450 ff ff ff ff ff ff ff ff ff ff ff ff ff ff 08 42 ...............B
125. 00000460 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
126. 00000470 08 42 08 42 08 42 08 42 08 42 08 42 08 42 ff ff .B.B.B.B.B.B.B..
127. 00000480 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
128. 00000490 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
129. 000004a0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
130. 000004b0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
131. 000004c0 ff ff 10 84 10 84 10 84 10 84 10 84 10 84 10 84 ................
132. 000004d0 10 84 10 84 10 84 10 84 10 84 10 84 10 84 08 42 ...............B
133. 000004e0 08 42 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
134. 000004f0 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
135. 00000500 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
136. 00000510 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
137. 00000520 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
138. 00000530 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
139. 00000540 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
140. 00000550 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
141. 00000560 08 42 10 84 99 d6 99 d6 99 d6 00 00 00 00 99 d6 .B..............
142. 00000570 99 d6 99 d6 99 d6 00 00 00 00 99 d6 99 d6 ff ff ................
143. 00000580 ff ff 99 d6 99 d6 99 d6 00 00 00 00 99 d6 99 d6 ................
144. 00000590 99 d6 99 d6 00 00 00 00 99 d6 99 d6 10 84 08 42 ...............B
145. 000005a0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 00 00 00 00 .B..............
146. 000005b0 99 d6 99 d6 00 00 00 00 99 d6 99 d6 99 d6 ff ff ................
147. 000005c0 ff ff 99 d6 99 d6 99 d6 99 d6 00 00 00 00 99 d6 ................
148. 000005d0 99 d6 00 00 00 00 99 d6 99 d6 99 d6 10 84 08 42 ...............B
149. 000005e0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 00 00 .B..............
150. 000005f0 00 00 00 00 00 00 99 d6 99 d6 99 d6 99 d6 ff ff ................
151. 00000600 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 00 00 00 00 ................
152. 00000610 00 00 00 00 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
153. 00000620 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
154. 00000630 00 00 00 00 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
155. 00000640 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 00 00 ................
156. 00000650 00 00 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
157. 00000660 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 00 00 .B..............
158. 00000670 00 00 00 00 00 00 99 d6 99 d6 99 d6 99 d6 ff ff ................
159. 00000680 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 00 00 00 00 ................
160. 00000690 00 00 00 00 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
161. 000006a0 08 42 10 84 99 d6 99 d6 99 d6 99 d6 00 00 00 00 .B..............
162. 000006b0 99 d6 99 d6 00 00 00 00 99 d6 99 d6 99 d6 ff ff ................
163. 000006c0 ff ff 99 d6 99 d6 99 d6 99 d6 00 00 00 00 99 d6 ................
164. 000006d0 99 d6 00 00 00 00 99 d6 99 d6 99 d6 10 84 08 42 ...............B
165. 000006e0 08 42 10 84 99 d6 99 d6 99 d6 00 00 00 00 99 d6 .B..............
166. 000006f0 99 d6 99 d6 99 d6 00 00 00 00 99 d6 99 d6 ff ff ................
167. 00000700 ff ff 99 d6 99 d6 99 d6 00 00 00 00 99 d6 99 d6 ................
168. 00000710 99 d6 99 d6 00 00 00 00 99 d6 99 d6 10 84 08 42 ...............B
169. 00000720 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
170. 00000730 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
171. 00000740 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
172. 00000750 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
173. 00000760 08 42 10 84 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 .B..............
174. 00000770 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ff ff ................
175. 00000780 ff ff 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 ................
176. 00000790 99 d6 99 d6 99 d6 99 d6 99 d6 99 d6 10 84 08 42 ...............B
177. 000007a0 08 42 10 84 10 84 10 84 10 84 10 84 10 84 10 84 .B..............
178. 000007b0 10 84 10 84 10 84 10 84 10 84 10 84 99 d6 ff ff ................
179. 000007c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
180. 000007d0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff 08 42 ...............B
181. 000007e0 08 42 08 42 08 42 08 42 08 42 08 42 08 42 08 42 .B.B.B.B.B.B.B.B
182. 000007f0 08 42 08 42 08 42 08 42 08 42 08 42 08 42 ff ff .B.B.B.B.B.B.B..

### Cache Color Table

The following is an annotated dump of a [Cache Color Table (section 2.2.2.2.1.2.4)](#Section_49fad30181714f38a68a7394baeb7b15) Secondary Drawing Order.

1. 00000000 03 fc 03 00 00 01 00 00 01 00 00 00 00 00 00 80 ................
2. 00000010 00 00 80 00 00 00 80 80 00 80 00 00 00 80 00 80 ................
3. 00000020 00 80 80 00 00 c0 c0 c0 00 c0 dc c0 00 f0 ca a6 ................
4. 00000030 00 01 1f 3f 00 01 1f 5f 00 01 1f 7f 00 01 1f 9f ...?...\_........
5. 00000040 00 01 1f bf 00 01 1f df 00 01 3f 01 00 01 3f 1f ..........?...?.
6. 00000050 00 01 3f 3f 00 01 3f 5f 00 01 3f 7f 00 01 3f 9f ..??..?\_..?...?.
7. 00000060 00 01 3f bf 00 01 3f df 00 01 5f 01 00 01 5f 1f ..?...?...\_...\_.
8. 00000070 00 01 5f 3f 00 01 5f 5f 00 01 5f 7f 00 01 5f 9f ..\_?..\_\_..\_...\_.
9. 00000080 00 01 5f bf 00 01 5f df 00 01 7f 01 00 01 7f 1f ..\_...\_.........
10. 00000090 00 01 7f 3f 00 01 7f 5f 00 01 7f 7f 00 01 7f 9f ...?...\_........
11. 000000a0 00 01 7f bf 00 01 7f df 00 01 9f 01 00 01 9f 1f ................
12. 000000b0 00 01 9f 3f 00 01 9f 5f 00 01 9f 7f 00 01 9f 9f ...?...\_........
13. 000000c0 00 01 9f bf 00 01 9f df 00 01 bf 01 00 01 bf 1f ................
14. 000000d0 00 01 bf 3f 00 01 bf 5f 00 01 bf 7f 00 01 bf 9f ...?...\_........
15. 000000e0 00 01 bf bf 00 01 bf df 00 01 df 01 00 01 df 1f ................
16. 000000f0 00 01 df 3f 00 01 df 5f 00 01 df 7f 00 01 df 9f ...?...\_........
17. 00000100 00 01 df bf 00 01 df df 00 3f 01 01 00 3f 01 1f .........?...?..
18. 00000110 00 3f 01 3f 00 3f 01 5f 00 3f 01 7f 00 3f 01 9f .?.?.?.\_.?...?..
19. 00000120 00 3f 01 bf 00 3f 01 df 00 3f 1f 01 00 3f 1f 1f .?...?...?...?..
20. 00000130 00 3f 1f 3f 00 3f 1f 5f 00 3f 1f 7f 00 3f 1f 9f .?.?.?.\_.?...?..
21. 00000140 00 3f 1f bf 00 3f 1f df 00 3f 3f 01 00 3f 3f 1f .?...?...??..??.
22. 00000150 00 3f 3f 3f 00 3f 3f 5f 00 3f 3f 7f 00 3f 3f 9f .???.??\_.??..??.
23. 00000160 00 3f 3f bf 00 3f 3f df 00 3f 5f 01 00 3f 5f 1f .??..??..?\_..?\_.
24. 00000170 00 3f 5f 3f 00 3f 5f 5f 00 3f 5f 7f 00 3f 5f 9f .?\_?.?\_\_.?\_..?\_.
25. 00000180 00 3f 5f bf 00 3f 5f df 00 3f 7f 01 00 3f 7f 1f .?\_..?\_..?...?..
26. 00000190 00 3f 7f 3f 00 3f 7f 5f 00 3f 7f 7f 00 3f 7f 9f .?.?.?.\_.?...?..
27. 000001a0 00 3f 7f bf 00 3f 7f df 00 3f 9f 01 00 3f 9f 1f .?...?...?...?..
28. 000001b0 00 3f 9f 3f 00 3f 9f 5f 00 3f 9f 7f 00 3f 9f 9f .?.?.?.\_.?...?..
29. 000001c0 00 3f 9f bf 00 3f 9f df 00 3f bf 01 00 3f bf 1f .?...?...?...?..
30. 000001d0 00 3f bf 3f 00 3f bf 5f 00 3f bf 7f 00 3f bf 9f .?.?.?.\_.?...?..
31. 000001e0 00 3f bf bf 00 3f bf df 00 3f df 01 00 3f df 1f .?...?...?...?..
32. 000001f0 00 3f df 3f 00 3f df 5f 00 3f df 7f 00 3f df 9f .?.?.?.\_.?...?..
33. 00000200 00 3f df bf 00 3f df df 00 7f 01 01 00 7f 01 1f .?...?..........
34. 00000210 00 7f 01 3f 00 7f 01 5f 00 7f 01 7f 00 7f 01 9f ...?...\_........
35. 00000220 00 7f 01 bf 00 7f 01 df 00 7f 1f 01 00 7f 1f 1f ................
36. 00000230 00 7f 1f 3f 00 7f 1f 5f 00 7f 1f 7f 00 7f 1f 9f ...?...\_........
37. 00000240 00 7f 1f bf 00 7f 1f df 00 7f 3f 01 00 7f 3f 1f ..........?...?.
38. 00000250 00 7f 3f 3f 00 7f 3f 5f 00 7f 3f 7f 00 7f 3f 9f ..??..?\_..?...?.
39. 00000260 00 7f 3f bf 00 7f 3f df 00 7f 5f 01 00 7f 5f 1f ..?...?...\_...\_.
40. 00000270 00 7f 5f 3f 00 7f 5f 5f 00 7f 5f 7f 00 7f 5f 9f ..\_?..\_\_..\_...\_.
41. 00000280 00 7f 5f bf 00 7f 5f df 00 7f 7f 01 00 7f 7f 1f ..\_...\_.........
42. 00000290 00 7f 7f 3f 00 7f 7f 5f 00 7f 7f 7f 00 7f 7f 9f ...?...\_........
43. 000002a0 00 7f 7f bf 00 7f 7f df 00 7f 9f 01 00 7f 9f 1f ................
44. 000002b0 00 7f 9f 3f 00 7f 9f 5f 00 7f 9f 7f 00 7f 9f 9f ...?...\_........
45. 000002c0 00 7f 9f bf 00 7f 9f df 00 7f bf 01 00 7f bf 1f ................
46. 000002d0 00 7f bf 3f 00 7f bf 5f 00 7f bf 7f 00 7f bf 9f ...?...\_........
47. 000002e0 00 7f bf bf 00 7f bf df 00 7f df 01 00 7f df 1f ................
48. 000002f0 00 7f df 3f 00 7f df 5f 00 7f df 7f 00 7f df 9f ...?...\_........
49. 00000300 00 7f df bf 00 7f df df 00 bf 01 01 00 bf 01 1f ................
50. 00000310 00 bf 01 3f 00 bf 01 5f 00 bf 01 7f 00 bf 01 9f ...?...\_........
51. 00000320 00 bf 01 bf 00 bf 01 df 00 bf 1f 01 00 bf 1f 1f ................
52. 00000330 00 bf 1f 3f 00 bf 1f 5f 00 bf 1f 7f 00 bf 1f 9f ...?...\_........
53. 00000340 00 bf 1f bf 00 bf 1f df 00 bf 3f 01 00 bf 3f 1f ..........?...?.
54. 00000350 00 bf 3f 3f 00 bf 3f 5f 00 bf 3f 7f 00 bf 3f 9f ..??..?\_..?...?.
55. 00000360 00 bf 3f bf 00 bf 3f df 00 bf 5f 01 00 bf 5f 1f ..?...?...\_...\_.
56. 00000370 00 bf 5f 3f 00 bf 5f 5f 00 bf 5f 7f 00 bf 5f 9f ..\_?..\_\_..\_...\_.
57. 00000380 00 bf 5f bf 00 bf 5f df 00 bf 7f 01 00 bf 7f 1f ..\_...\_.........
58. 00000390 00 bf 7f 3f 00 bf 7f 5f 00 bf 7f 7f 00 bf 7f 9f ...?...\_........
59. 000003a0 00 bf 7f bf 00 bf 7f df 00 bf 9f 01 00 bf 9f 1f ................
60. 000003b0 00 bf 9f 3f 00 bf 9f 5f 00 bf 9f 7f 00 bf 9f 9f ...?...\_........
61. 000003c0 00 bf 9f bf 00 bf 9f df 00 bf bf 01 00 bf bf 1f ................
62. 000003d0 00 bf bf 3f 00 bf bf 5f 00 bf bf 7f 00 bf bf 9f ...?...\_........
63. 000003e0 00 f0 fb ff 00 a4 a0 a0 00 80 80 80 00 00 00 ff ................
64. 000003f0 00 00 ff 00 00 00 ff ff 00 ff 00 00 00 ff 00 ff ................
65. 00000400 00 ff ff 00 00 ff ff ff 00 ................
66. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
67. 0x03 =
68. 0x01 |
69. 0x02
70. = TS\_STANDARD |
71. TS\_SECONDARY
72. fc 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0x3fc + 13 = 1033 bytes
73. 00 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = 0
74. 01 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = TS\_CACHE\_COLOR\_TABLE
75. 00 -> CACHE\_COLOR\_TABLE\_ORDER::cacheIndex
76. 00 01 -> CACHE\_COLOR\_TABLE\_ORDER::numberColors = 0x0100 = 256 Colors
77. CACHE\_COLOR\_TABLE\_ORDER::ColorTable:
78. R = red
79. G = green
80. B = blue
81. P = padding
82. Index R G B P
83. 0000 00 00 00 00
84. 0004 00 00 80 00
85. 0008 00 80 00 00
86. 000c 00 80 80 00
87. 0010 80 00 00 00
88. 0014 80 00 80 00
89. 0018 80 80 00 00
90. 001c c0 c0 c0 00
91. 0020 c0 dc c0 00
92. 0024 f0 ca a6 00
93. 0028 01 1f 3f 00
94. 002c 01 1f 5f 00
95. 0030 01 1f 7f 00
96. 0034 01 1f 9f 00
97. 0038 01 1f bf 00
98. 003c 01 1f df 00
99. 0040 01 3f 01 00
100. 0044 01 3f 1f 00
101. 0048 01 3f 3f 00
102. 004c 01 3f 5f 00
103. 0050 01 3f 7f 00
104. 0054 01 3f 9f 00
105. 0058 01 3f bf 00
106. 005c 01 3f df 00
107. 0060 01 5f 01 00
108. 0064 01 5f 1f 00
109. 0068 01 5f 3f 00
110. 006c 01 5f 5f 00
111. 0070 01 5f 7f 00
112. 0074 01 5f 9f 00
113. 0078 01 5f bf 00
114. 007c 01 5f df 00
115. 0080 01 7f 01 00
116. 0084 01 7f 1f 00
117. 0088 01 7f 3f 00
118. 008c 01 7f 5f 00
119. 0090 01 7f 7f 00
120. 0094 01 7f 9f 00
121. 0098 01 7f bf 00
122. 009c 01 7f df 00
123. 00a0 01 9f 01 00
124. 00a4 01 9f 1f 00
125. 00a8 01 9f 3f 00
126. 00ac 01 9f 5f 00
127. 00b0 01 9f 7f 00
128. 00b4 01 9f 9f 00
129. 00b8 01 9f bf 00
130. 00bc 01 9f df 00
131. 00c0 01 bf 01 00
132. 00c4 01 bf 1f 00
133. 00c8 01 bf 3f 00
134. 00cc 01 bf 5f 00
135. 00d0 01 bf 7f 00
136. 00d4 01 bf 9f 00
137. 00d8 01 bf bf 00
138. 00dc 01 bf df 00
139. 00e0 01 df 01 00
140. 00e4 01 df 1f 00
141. 00e8 01 df 3f 00
142. 00ec 01 df 5f 00
143. 00f0 01 df 7f 00
144. 00f4 01 df 9f 00
145. 00f8 01 df bf 00
146. 00fc 01 df df 00
147. 0100 3f 01 01 00
148. 0104 3f 01 1f 00
149. 0108 3f 01 3f 00
150. 010c 3f 01 5f 00
151. 0110 3f 01 7f 00
152. 0114 3f 01 9f 00
153. 0118 3f 01 bf 00
154. 011c 3f 01 df 00
155. 0120 3f 1f 01 00
156. 0124 3f 1f 1f 00
157. 0128 3f 1f 3f 00
158. 012c 3f 1f 5f 00
159. 0130 3f 1f 7f 00
160. 0134 3f 1f 9f 00
161. 0138 3f 1f bf 00
162. 013c 3f 1f df 00
163. 0140 3f 3f 01 00
164. 0144 3f 3f 1f 00
165. 0148 3f 3f 3f 00
166. 014c 3f 3f 5f 00
167. 0150 3f 3f 7f 00
168. 0154 3f 3f 9f 00
169. 0158 3f 3f bf 00
170. 015c 3f 3f df 00
171. 0160 3f 5f 01 00
172. 0164 3f 5f 1f 00
173. 0168 3f 5f 3f 00
174. 016c 3f 5f 5f 00
175. 0170 3f 5f 7f 00
176. 0174 3f 5f 9f 00
177. 0178 3f 5f bf 00
178. 017c 3f 5f df 00
179. 0180 3f 7f 01 00
180. 0184 3f 7f 1f 00
181. 0188 3f 7f 3f 00
182. 018c 3f 7f 5f 00
183. 0190 3f 7f 7f 00
184. 0194 3f 7f 9f 00
185. 0198 3f 7f bf 00
186. 019c 3f 7f df 00
187. 01a0 3f 9f 01 00
188. 01a4 3f 9f 1f 00
189. 01a8 3f 9f 3f 00
190. 01ac 3f 9f 5f 00
191. 01b0 3f 9f 7f 00
192. 01b4 3f 9f 9f 00
193. 01b8 3f 9f bf 00
194. 01bc 3f 9f df 00
195. 01c0 3f bf 01 00
196. 01c4 3f bf 1f 00
197. 01c8 3f bf 3f 00
198. 01cc 3f bf 5f 00
199. 01d0 3f bf 7f 00
200. 01d4 3f bf 9f 00
201. 01d8 3f bf bf 00
202. 01dc 3f bf df 00
203. 01e0 3f df 01 00
204. 01e4 3f df 1f 00
205. 01e8 3f df 3f 00
206. 01ec 3f df 5f 00
207. 01f0 3f df 7f 00
208. 01f4 3f df 9f 00
209. 01f8 3f df bf 00
210. 01fc 3f df df 00
211. 0200 7f 01 01 00
212. 0204 7f 01 1f 00
213. 0208 7f 01 3f 00
214. 020c 7f 01 5f 00
215. 0210 7f 01 7f 00
216. 0214 7f 01 9f 00
217. 0218 7f 01 bf 00
218. 021c 7f 01 df 00
219. 0220 7f 1f 01 00
220. 0224 7f 1f 1f 00
221. 0228 7f 1f 3f 00
222. 022c 7f 1f 5f 00
223. 0230 7f 1f 7f 00
224. 0234 7f 1f 9f 00
225. 0238 7f 1f bf 00
226. 023c 7f 1f df 00
227. 0240 7f 3f 01 00
228. 0244 7f 3f 1f 00
229. 0248 7f 3f 3f 00
230. 024c 7f 3f 5f 00
231. 0250 7f 3f 7f 00
232. 0254 7f 3f 9f 00
233. 0258 7f 3f bf 00
234. 025c 7f 3f df 00
235. 0260 7f 5f 01 00
236. 0264 7f 5f 1f 00
237. 0268 7f 5f 3f 00
238. 026c 7f 5f 5f 00
239. 0270 7f 5f 7f 00
240. 0274 7f 5f 9f 00
241. 0278 7f 5f bf 00
242. 027c 7f 5f df 00
243. 0280 7f 7f 01 00
244. 0284 7f 7f 1f 00
245. 0288 7f 7f 3f 00
246. 028c 7f 7f 5f 00
247. 0290 7f 7f 7f 00
248. 0294 7f 7f 9f 00
249. 0298 7f 7f bf 00
250. 029c 7f 7f df 00
251. 02a0 7f 9f 01 00
252. 02a4 7f 9f 1f 00
253. 02a8 7f 9f 3f 00
254. 02ac 7f 9f 5f 00
255. 02b0 7f 9f 7f 00
256. 02b4 7f 9f 9f 00
257. 02b8 7f 9f bf 00
258. 02bc 7f 9f df 00
259. 02c0 7f bf 01 00
260. 02c4 7f bf 1f 00
261. 02c8 7f bf 3f 00
262. 02cc 7f bf 5f 00
263. 02d0 7f bf 7f 00
264. 02d4 7f bf 9f 00
265. 02d8 7f bf bf 00
266. 02dc 7f bf df 00
267. 02e0 7f df 01 00
268. 02e4 7f df 1f 00
269. 02e8 7f df 3f 00
270. 02ec 7f df 5f 00
271. 02f0 7f df 7f 00
272. 02f4 7f df 9f 00
273. 02f8 7f df bf 00
274. 02fc 7f df df 00
275. 0300 bf 01 01 00
276. 0304 bf 01 1f 00
277. 0308 bf 01 3f 00
278. 030c bf 01 5f 00
279. 0310 bf 01 7f 00
280. 0314 bf 01 9f 00
281. 0318 bf 01 bf 00
282. 031c bf 01 df 00
283. 0320 bf 1f 01 00
284. 0324 bf 1f 1f 00
285. 0328 bf 1f 3f 00
286. 032c bf 1f 5f 00
287. 0330 bf 1f 7f 00
288. 0334 bf 1f 9f 00
289. 0338 bf 1f bf 00
290. 033c bf 1f df 00
291. 0340 bf 3f 01 00
292. 0344 bf 3f 1f 00
293. 0348 bf 3f 3f 00
294. 034c bf 3f 5f 00
295. 0350 bf 3f 7f 00
296. 0354 bf 3f 9f 00
297. 0358 bf 3f bf 00
298. 035c bf 3f df 00
299. 0360 bf 5f 01 00
300. 0364 bf 5f 1f 00
301. 0368 bf 5f 3f 00
302. 036c bf 5f 5f 00
303. 0370 bf 5f 7f 00
304. 0374 bf 5f 9f 00
305. 0378 bf 5f bf 00
306. 037c bf 5f df 00
307. 0380 bf 7f 01 00
308. 0384 bf 7f 1f 00
309. 0388 bf 7f 3f 00
310. 038c bf 7f 5f 00
311. 0390 bf 7f 7f 00
312. 0394 bf 7f 9f 00
313. 0398 bf 7f bf 00
314. 039c bf 7f df 00
315. 03a0 bf 9f 01 00
316. 03a4 bf 9f 1f 00
317. 03a8 bf 9f 3f 00
318. 03ac bf 9f 5f 00
319. 03b0 bf 9f 7f 00
320. 03b4 bf 9f 9f 00
321. 03b8 bf 9f bf 00
322. 03bc bf 9f df 00
323. 03c0 bf bf 01 00
324. 03c4 bf bf 1f 00
325. 03c8 bf bf 3f 00
326. 03cc bf bf 5f 00
327. 03d0 bf bf 7f 00
328. 03d4 bf bf 9f 00
329. 03d8 f0 fb ff 00
330. 03dc a4 a0 a0 00
331. 03e0 80 80 80 00
332. 03e4 00 00 ff 00
333. 03e8 00 ff 00 00
334. 03ec 00 ff ff 00
335. 03f0 ff 00 00 00
336. 03f4 ff 00 ff 00
337. 03f8 ff ff 00 00
338. 03fc ff ff ff 00

### Cache Glyph (Revision 1)

The following is an annotated dump of a Cache Glyph - Revision 1 (section [2.2.2.2.1.2.5)](#Section_6dc06088b1124f4fb0bcb97bba621a0c) Secondary Drawing Order.

1. 00000000 03 37 00 10 00 03 07 03 31 00 00 00 f8 ff 05 00 .7......1.......
2. 00000010 08 00 c0 c0 c0 c0 c0 c0 c0 f8 32 00 00 00 fa ff ..........2.....
3. 00000020 06 00 08 00 7c cc cc cc cc 7c 0c 78 33 00 00 00 ....|....|.x3...
4. 00000030 f8 ff 07 00 08 00 7c c6 c6 c6 c6 c6 c6 7c 4c 00 ......|......|L.
5. 00000040 67 00 4f 00 g.O.
6. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
7. 0x03
8. = 0x2 |
9. 0x1
10. = TS\_SECONDARY |
11. TS\_STANDARD
12. 37 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0x0037 + 13 = 68 bytes
13. 10 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = CG\_GLYPH\_UNICODE\_PRESENT (0x0010)
14. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = 0x03 = TS\_CACHE\_GLYPH
15. 07 -> CACHE\_GLYPH\_ORDER::cacheId = 0x07
16. 03 -> CACHE\_GLYPH\_ORDER::cGlyphs = 0x03
17. 31 00 -> TS\_CACHE\_GLYPH\_DATA::cacheIndex = 0x0031 = 49
18. 00 00 -> TS\_CACHE\_GLYPH\_DATA::x = 0
19. f8 ff -> TS\_CACHE\_GLYPH\_DATA::y = 0xfff8 = -8
20. 05 00 -> TS\_CACHE\_GLYPH\_DATA::cx = 0x0005 = 5
21. 08 00 -> TS\_CACHE\_GLYPH\_DATA::cy = 0x0008 = 8
22. c0 -> XX...
23. c0 -> XX...
24. c0 -> XX...
25. c0 -> XX...
26. c0 -> XX...
27. c0 -> XX...
28. c0 -> XX...
29. f8 -> XXXXX
30. 32 00 -> TS\_CACHE\_GLYPH\_DATA::cacheIndex = 0x0032 = 50
31. 00 00 -> TS\_CACHE\_GLYPH\_DATA::x = 0
32. fa ff -> TS\_CACHE\_GLYPH\_DATA::y = 0xfffa = -6
33. 06 00 -> TS\_CACHE\_GLYPH\_DATA::cx = 0x0006 = 6
34. 08 00 -> TS\_CACHE\_GLYPH\_DATA::cy = 0x0008 = 8
35. 7c -> .XXXXX
36. cc -> XX..XX
37. cc -> XX..XX
38. cc -> XX..XX
39. cc -> XX..XX
40. 7c -> .XXXXX
41. 0c -> ....XX
42. 78 -> .XXXX.
43. 33 00 -> TS\_CACHE\_GLYPH\_DATA::cacheIndex = 0x0033 = 51
44. 00 00 -> TS\_CACHE\_GLYPH\_DATA::x = 0
45. f8 ff -> TS\_CACHE\_GLYPH\_DATA::y = 0xfff8 = -8
46. 07 00 -> TS\_CACHE\_GLYPH\_DATA::cx = 0x0007 = 7
47. 08 00 -> TS\_CACHE\_GLYPH\_DATA::cy = 0x0008 = 8
48. 7c -> .XXXXX.
49. c6 -> XX...XX
50. c6 -> XX...XX
51. c6 -> XX...XX
52. c6 -> XX...XX
53. c6 -> XX...XX
54. c6 -> XX...XX
55. 7c -> .XXXXX.
56. 4c 00 67 00 4f 00 -> CACHE\_GLYPH\_ORDER::unicodeCharacters = 0x004c 0x0067 0x004f = LgO

### Cache Glyph (Revision 2)

The following is an annotated dump of a Cache Glyph - Revision 2 (section [2.2.2.2.1.2.6)](#Section_ffc80435136e442e908540ca1d27a413) Secondary Drawing Order.

1. 00000000 03 04 00 37 01 03 34 00 49 03 03 a0 a0 a0 00 22 ...7..4.I......"
2. 00000010 00 .
3. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
4. 0x03
5. = 0x2 |
6. 0x1
7. = TS\_SECONDARY |
8. TS\_STANDARD
9. 04 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0x04 + 13 = 17 bytes
10. 37 01 -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = 0x137 = binary:00000001 0011 0111
11. cacheId = binary:0111 = 7
12. flags = binary:0011 = 0x03
13. 0x03
14. = 0x01 |
15. 0x02
16. = CG2\_GLYPH\_UNICODE\_PRESENT |
17. GLYPH\_ORDER\_REV2
19. cGlyphs = 0000 0001 = 0x1 = 1
20. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = 0x03 = TS\_CACHE\_GLYPH
21. 34 -> TS\_CACHE\_GLYPH\_DATA\_REV2::cacheIndex = 0x34 = 52
22. 00 -> TS\_CACHE\_GLYPH\_DATA\_REV2::x = 0
23. 49 -> TS\_CACHE\_GLYPH\_DATA\_REV2::y = -9
24. 03 -> TS\_CACHE\_GLYPH\_DATA\_REV2::cx = 3
25. 03 -> TS\_CACHE\_GLYPH\_DATA\_REV2::cy = 3
26. a0 a0 a0 00 -> TS\_CACHE\_GLYPH\_DATA\_REV2::aj
27. a0 -> X.X
28. a0 -> X.X
29. a0 -> X.X
30. 00 -> padding
31. 22 00 -> CACHE\_GLYPH\_REV2\_ORDER::unicodeCharacters = 0x0022 = "

### Cache Brush

The following is an annotated dump of a [Cache Brush (section 2.2.2.2.1.2.7)](#Section_9e7027c4d9894cb09674e2e36c95ebdd) Secondary Drawing Order.

1. 00000000 03 07 00 00 00 07 00 01 08 08 81 08 aa 55 aa 55 .............U.U
2. 00000010 aa 55 aa 55 .U.U
3. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
4. 0x03
5. = 0x2 |
6. 0x1
7. = TS\_SECONDARY |
8. TS\_STANDARD
9. 07 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0x7 + 13 = 20 bytes
10. 00 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = 0x0
11. 07 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = 0x07 = TS\_CACHE\_BRUSH
12. 00 -> CACHE\_BRUSH\_ORDER::cacheEntry = 0
13. 01 -> CACHE\_BRUSH\_ORDER::iBitmapFormat = TS\_BMF\_1BPP (0x01)
14. 08 -> CACHE\_BRUSH\_ORDER::cx = 0x8 = 8
15. 08 -> CACHE\_BRUSH\_ORDER::cy = 0x8 = 8
16. 81 -> CACHE\_BRUSH\_ORDER::Style = 0x81
17. 08 -> CACHE\_BRUSH\_ORDER::iBytes = 0x8 = 8 bytes
18. aa 55 aa 55 aa 55 aa 55 -> CACHE\_BRUSH\_ORDER::brushData
19. 0xaa -> 1 0 1 0 1 0 1 0
20. 0x55 -> 0 1 0 1 0 1 0 1
21. 0xaa -> 1 0 1 0 1 0 1 0
22. 0x55 -> 0 1 0 1 0 1 0 1
23. 0xaa -> 1 0 1 0 1 0 1 0
24. 0x55 -> 0 1 0 1 0 1 0 1
25. 0xaa -> 1 0 1 0 1 0 1 0
26. 0x55 -> 0 1 0 1 0 1 0 1

### Cache Bitmap (Revision 3)

The following is an annotated dump of a Cache Bitmap - Revision 3 (section [2.2.2.2.1.2.8)](#Section_710cf5c26a0c4cea8d62fa4f639d7c5a) secondary drawing order.

1. 00000000 03 37 00 30 0c 08 ff 7f 35 50 ec bc 74 52 65 b7 .7.0....5P..tRe.
2. 00000010 20 00 00 00 05 00 02 00 28 00 00 00 5b 4f 45 ff .......(...[OE.
3. 00000020 5b 4f 45 ff 5b 4f 45 ff 5b 4f 45 ff 5b 4f 45 ff [OE.[OE.[OE.[OE.
4. 00000030 5b 50 45 ff 5b 50 45 ff 5b 50 45 ff 5b 50 45 ff [PE.[PE.[PE.[PE.
5. 00000040 5b 50 45 ff [PE.
6. 03 -> SECONDARY\_DRAWING\_ORDER\_HEADER::controlFlags = 0x03
7. 0x03
8. = 0x2 |
9. 0x1
10. = TS\_SECONDARY |
11. TS\_STANDARD
12. 37 00 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderLength = 0x37 + 13 = 68 bytes
13. 30 0c -> SECONDARY\_DRAWING\_ORDER\_HEADER::extraFlags = 0x0c30 = binary:11000 0110 000
14. cacheId = binary:000 = 0
15. bitsPerPixelID = binary:0110 = 0x6 (32 bpp)
16. flags = binary:11000 = 0x18
17. 0x18
18. = 0x10 |
19. 0x08 |
20. = CBR3\_DO\_NOT\_CACHE |
21. CBR3\_IGNORABLE\_FLAG
22. 08 -> SECONDARY\_DRAWING\_ORDER\_HEADER::orderType = 0x08 = TS\_CACHE\_BITMAP\_COMPRESSED\_REV3
23. ff 7f -> CACHE\_BITMAP\_REV3\_ORDER::cacheIndex = 0x7fff = 32767
24. 35 50 ec bc -> CACHE\_BITMAP\_REV3\_ORDER::key1 = 0xbcec5035
25. 74 52 65 b7 -> CACHE\_BITMAP\_REV3\_ORDER::key2 = 0xb7655274
26. 20 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::bpp = 0x20 = 32 bpp
27. 00 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::reserved1
28. 00 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::reserved2
29. 00 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::codecID = 0x00 (uncompressed)
30. 05 00 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::width = 0x0005 = 5
31. 02 00 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::height = 0x0002 = 2
32. 28 00 00 00 -> CACHE\_BITMAP\_REV3\_ORDER::bitmapData::bitmapDataLength = 0x00000028 = 40 bytes
33. Bitmap data (40 bytes):
34. 5b 4f 45 ff 5b 4f 45 ff 5b 4f 45 ff 5b 4f 45 ff
35. 5b 4f 45 ff 5b 50 45 ff 5b 50 45 ff 5b 50 45 ff
36. 5b 50 45 ff 5b 50 45 ff

## Annotated Alternate Secondary Drawing Orders

### Create Offscreen Bitmap

The following is an annotated dump of a [Create Offscreen Bitmap (section 2.2.2.2.1.3.2)](#Section_427758ff4bac4833b388d634cc512d0c) Alternate Secondary Drawing Order.

1. 00000000 06 00 80 60 01 10 00 01 00 02 00 ...`.......
2. 06 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x06 = binary:000001 10
3. class = TS\_SECONDARY (0x02)
4. orderType = TS\_ALTSEC\_CREATE\_OFFSCR\_BITMAP (0x01)
5. 00 80 -> CREATE\_OFFSCR\_BITMAP\_ORDER::flags = 0x8000
6. OffscreenBitmapId = 0
7. d = 1
8. 60 01 -> CREATE\_OFFSCR\_BITMAP\_ORDER::cx = 0x0160 = 352
9. 10 00 -> CREATE\_OFFSCR\_BITMAP\_ORDER::cy = 0x0010 = 16
10. 01 00 -> OFFSCR\_DELETE\_LIST::cIndices = 1
11. 02 00 -> OFFSCR\_DELETE\_LIST::indices[0] = 0x0002

### Switch Surface

The following is an annotated dump of a [Switch Surface (section 2.2.2.2.1.3.3)](#Section_ada4d8a9421e48c0a0ed7384fa3cf061) Alternate Secondary Drawing Order.

1. 00000000 02 ff ff ...
2. 02 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x02 = binary:000000 10
3. class = TS\_SECONDARY (0x02)
4. orderType = TS\_ALTSEC\_SWITCH\_SURFACE (0x00)
5. ff ff -> SWITCH\_SURFACE\_ORDER::bitmapId = SCREEN\_BITMAP\_SURFACE (0xFFFF)

### Create NineGrid Bitmap

The following is an annotated dump of a [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38) Alternate Secondary Drawing Order.

1. 00000000 12 20 41 00 08 00 ab 01 05 00 00 00 00 00 07 00 . A.............
2. 00000010 07 00 02 00 01 00 00 00 ........
3. 12 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x12
4. class = TS\_SECONDARY (0x02)
5. orderType = TS\_ALTSEC\_CREATE\_NINEGRID\_BITMAP (0x04)
6. 20 -> CREATE\_NINEGRID\_BITMAP\_ORDER::BitmapBpp = 0x20 = 32 Bpp
7. 41 00 -> CREATE\_NINEGRID\_BITMAP\_ORDER::BitmapId = 0x41 = 65
8. 08 00 -> CREATE\_NINEGRID\_BITMAP\_ORDER::cx = 0x8 = 8
9. ab 01 -> CREATE\_NINEGRID\_BITMAP\_ORDER::cy = 0x01ab = 427
10. 05 00 00 00 -> NINEGRID\_BITMAP\_INFO::flFlags = 0x5
11. 0x5
12. = 0x1 |
13. 0x4
14. = DSDNG\_STRETCH |
15. DSDNG\_PERPIXELALPHA
16. 00 00 -> NINEGRID\_BITMAP\_INFO::ulLeftWidth = 0x0 = 0
17. 07 00 -> NINEGRID\_BITMAP\_INFO::ulRightWidth = 0x7 = 7
18. 07 00 -> NINEGRID\_BITMAP\_INFO::ulTopHeight = 0x7 = 7
19. 02 00 -> NINEGRID\_BITMAP\_INFO::ulBottomHeight = 0x2 = 2
20. 01 00 00 00 -> NINEGRID\_BITMAP\_INFO::crTransparent = 0x00000001
21. 01 -> TS\_COLORREF::red = 0x1 = 1
22. 00 -> TS\_COLORREF::green = 0
23. 00 -> TS\_COLORREF::blue = 0
24. 00 -> TS\_COLORREF::zeroPad = 0

### Stream Bitmap First

The following is an annotated dump of a [Stream Bitmap First (section 2.2.2.2.1.3.5.1)](#Section_b714a7c97fd54621a397349a4ef3e06c) Alternate Secondary Drawing Order.

1. 00000000 0a 07 20 01 00 04 00 1e 00 7a 01 00 00 7a 01 02 .. ......z...z..
2. 00000010 1c df 60 02 90 df 60 02 fc 19 f5 6c 8f 6a 9a 02 ..`...`....l.j..
3. 00000020 5f 07 00 00 88 df 60 02 02 00 00 00 a8 e6 8d 02 \_.....`.........
4. 00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
5. 00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
6. 00000050 00 00 00 00 01 00 00 00 01 00 00 00 84 df 60 02 ..............`.
7. 00000060 6f e3 28 e7 08 00 00 00 05 40 00 80 00 00 00 00 o.(......@......
8. 00000070 ee 71 9a 02 a8 e0 60 02 e3 57 ec 6c 02 00 00 00 .q....`..W.l....
9. 00000080 8f 6a 9a 02 5f 07 00 00 90 e0 60 02 04 00 00 00 .j..\_.....`.....
10. 00000090 05 73 9a 02 1f e7 99 02 1e 00 00 00 01 00 00 00 .s..............
11. 000000a0 a8 e6 8d 02 01 00 00 00 d0 df 60 02 02 00 00 00 ..........`.....
12. 000000b0 fc c3 38 00 ff 17 00 00 10 2f 99 02 0f 73 9a 02 ..8....../...s..
13. 000000c0 00 00 00 00 f0 2e 99 02 ba ed 7c 02 00 00 00 01 ..........|.....
14. 000000d0 8d 6a 9a 02 ff ff 00 00 00 00 00 00 00 2f 99 02 .j.........../..
15. 000000e0 48 52 f8 00 0d 00 00 00 00 00 00 00 04 00 00 00 HR..............
16. 000000f0 ba ed 7c 02 00 00 00 00 0a 00 00 00 00 01 00 00 ..|.............
17. 00000100 0a 00 00 00 10 33 99 02 ee 71 9a 02 10 73 99 02 .....3...q...s..
18. 00000110 ff 1f 00 00 ff ff 00 00 ff ff 00 00 68 e0 60 02 ............h.`.
19. 00000120 1a 4c ed 6c d0 ec 7c 02 ea 00 00 00 00 00 00 00 .L.l..|.........
20. 00000130 b8 e1 60 02 a8 e1 60 02 f0 2e 99 02 02 00 00 00 ..`...`.........
21. 00000140 02 00 00 00 04 00 00 00 87 dc 28 e7 b0 e0 60 02 ..........(...`.
22. 00000150 01 62 e5 6c 08 00 00 00 04 00 00 00 00 00 00 00 .b.l............
23. 00000160 08 00 00 00 8f 6a 9a 02 8f 6a 9a 02 5f 07 00 00 .....j...j..\_...
24. 00000170 00 00 00 00 c8 d8 7a 02 ee 71 9a 02 00 00 00 00 ......z..q......
25. 00000180 00 00 00 00 4b dc 28 e7 02 ....K.(..
26. 0a -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0xa
27. class = TS\_SECONDARY (0x02)
28. orderType = TS\_ALTSEC\_STREAM\_BITMAP\_FIRST (0x02)
29. 07 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapFlags = 0x07
30. 0x07
31. = 0x01 |
32. 0x02 |
33. 0x04
34. = STREAM\_BITMAP\_END |
35. STREAM\_BITMAP\_COMPRESSED |
36. STREAM\_BITMAP\_REV2
37. 20 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapBpp = 0x20 = 32 bpp
38. 01 00 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapType = TS\_DRAW\_NINEGRID\_BITMAP\_CACHE (0x1)
39. 04 00 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapWidth = 0x4 = 4
40. 1e 00 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapHeight = 0x1e = 30
41. 7a 01 00 00 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapSize = 0x17a = 378 bytes
42. 7a 01 -> STREAM\_BITMAP\_FIRST\_ORDER::BitmapBlockSize = 0x17a = 378 bytes
43. The next 378 bytes are the compressed bitmap (STREAM\_BITMAP\_FIRST\_ORDER::BitmapBlock).

### Stream Bitmap Next

The following is an annotated dump of a [Stream Bitmap Next (section 2.2.2.2.1.3.5.2)](#Section_dfe099ffa894406393a7fbe64e7cb85e) Alternate Secondary Drawing Order.

1. 00000000 0e 02 01 00 00 10 ff 03 71 f8 f8 f8 ff 03 c0 01 ........q.......
2. 00000010 12 12 12 00 03 c0 01 1b 1b 1b 00 03 c0 01 1e 1e ................
3. 00000020 1e 00 03 85 ed ed ed ff ed ed ed ff ed ed ed ff ................
4. 00000030 ed ed ed ff ed ed ed ff cc 01 01 01 00 00 97 6c ...............l
5. 00000040 eb eb eb ff 8d ec ......
6. ...
7. 00000fc0 ff fe fd fb ff fe fd fb ff fe fd fb ff fe fd fb ................
8. 00000fd0 ff fe fd fb ff fe fc fa ff eb e2 d5 ff b8 8f 59 ...............Y
9. 00000fe0 ff ea ea ea ff 00 00 00 00 00 00 00 00 00 00 00 ................
10. 00000ff0 00 f8 f8 f8 ff d3 c3 ae ff bf 9a 69 ff 6b b1 7f ...........i.k..
11. 00001000 3c ff 95 bc 93 5d <....]
12. 0e -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x0e
13. class = TS\_SECONDARY (0x02)
14. orderType = TS\_ALTSEC\_STREAM\_BITMAP\_NEXT (0x03)
15. 02 -> STREAM\_BITMAP\_NEXT\_ORDER::BitmapFlags = 0x2 = STREAM\_BITMAP\_COMPRESSED
16. 01 00 -> STREAM\_BITMAP\_NEXT\_ORDER::BitmapType = TS\_DRAW\_NINEGRID\_BITMAP\_CACHE (0x1)
17. 00 10 -> STREAM\_BITMAP\_NEXT\_ORDER::BitmapBlockSize = 0x1000 = 4096 bytes
18. The next 4096 bytes are the compressed bitmap (STREAM\_BITMAP\_FIRST\_ORDER::BitmapBlock).

### Draw GDI+ Cache First

The following sections provide examples of annotated dumps of [Draw GDI+ Cache First (section 2.2.2.2.1.3.6.2)](#Section_53dbc9c54e3f4b17a256baf43c582146) Alternate Secondary Drawing Orders.

#### Example 1

The following is the first example of an annotated dump of a [Draw GDI+ Cache First (section 2.2.2.2.1.3.6.2)](#Section_53dbc9c54e3f4b17a256baf43c582146) Alternate Secondary Drawing Order.

1. 00000000 22 00 04 00 00 00 00 10 1c 22 00 00 02 10 c0 db "........"......
2. 00000010 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3. 00000020 00 00 00 00 01 00 00 00 42 4d fe 21 00 00 00 00 ........BM.!....
4. 00000030 00 00 3e 00 00 00 28 00 00 00 08 01 00 00 f0 00 ..>...(.........
5. 00000040 00 00 01 00 01 00 00 00 00 00 00 00 00 00 00 00 ................
6. 00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
7. 00000060 00 00 ff ff ff 00 ff ff ff ff ff ff ff ff ff ff ................
8. 00000070 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
9. 00000080 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
10. 00000090 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
11. 000000a0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
12. 000000b0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
13. 000000c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
14. 000000d0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
15. 000000e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
16. 000000f0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
17. 00000100 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
18. 00000110 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
19. 00000120 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
20. 00000130 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
21. 00000140 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
22. 00000150 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
23. 00000160 00 00 ff ff ff ff d1 ff ff ff ff ff ff ff ff ff ................
24. 00000170 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
25. 00000180 ff ff ff 00 00 00 ff ff ff ff 80 7f ff ff ff ff ................
26. 00000190 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
27. 000001a0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff 7e 7f ..............~.
28. 000001b0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
29. 000001c0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
30. 000001d0 ff fe fe 3f ff ff ff ff ff ff ff ff ff ff ff ff ...?............
31. 000001e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
32. 000001f0 00 00 ff ff ff fd fe 1f ff ff ff ff ff ff ff ff ................
33. 00000200 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
34. 00000210 ff ff ff 00 00 00 ff ff ff e3 ff 1f ff ff ff ff ................
35. 00000220 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
36. 00000230 ff ff ff ff ff ff ff 00 00 00 ff ff ff e5 ff 8f ................
37. 00000240 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
38. 00000250 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
39. 00000260 ff c3 ff c7 ff ff ff ff ff ff ff ff ff ff fc 2f .............../
40. 00000270 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
41. 00000280 00 00 ff ff ff c7 ff f3 ff ff ff ff ff ff ff ff ................
42. 00000290 ff ff fd 0f ff ff ff ff ff ff ff ff ff ff ff ff ................
43. 000002a0 ff ff ff 00 00 00 ff ff ff c7 fe f9 ff ff ff ff ................
44. 000002b0 ff ff ff ff ff ff fe 9f ff ff ff ff ff ff ff ff ................
45. 000002c0 ff ff ff ff ff ff ff 00 00 00 ff ff ff c7 f9 fd ................
46. 000002d0 ff ff ff ff ff ff ff ff ff ff fe cf ff ff ff ff ................
47. 000002e0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
48. 000002f0 ff c1 f7 fd ff ff ff ff ff ff ff ff ff ff fe 8f ................
49. 00000300 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
50. 00000310 00 00 ff ff ff fd cf ff ff ff ff ff ff ff ff ff ................
51. 00000320 ff ff fe c7 ff ff ff ff ff ff ff ff ff ff ff ff ................
52. 00000330 ff ff ff 00 00 00 ff ff ff e0 1f fe ff ff ff ff ................
53. 00000340 ff ff ff ff ff ff fe ef ff ff ff ff ff ff ff ff ................
54. 00000350 ff ff ff ff ff ff ff 00 00 00 ff ff ff f8 ff fe ................
55. 00000360 ff ff ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
56. 00000370 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
57. 00000380 ff fd ff fe ff ff ff ff ff ff ff ff ff ff fe ef ................
58. 00000390 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
59. 000003a0 00 00 ff ff ff fc ff de ff fb ff ff ff ff ff ff ................
60. 000003b0 ff ff fe ef ff ff ff ff ff ff ff ff ff ff ff ff ................
61. 000003c0 ff ff ff 00 00 00 ff ff ff fe ff fe ff e3 ff ff ................
62. 000003d0 ff ff ff ff ff ff fe ef ff ff ff ff ff ff ff ff ................
63. 000003e0 ff ff ff ff ff ff ff 00 00 00 ff ff f3 ff ff fd ................
64. 000003f0 ff c3 ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
65. 00000400 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
66. 00000410 ef ff bf fd ff 8b ff ff ff ff ff ff ff ff fe ef ................
67. 00000420 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
68. 00000430 00 00 ff ff cf ff bf fd fc 38 ff ff ff ff ff ff .........8......
69. 00000440 ff ff fe 4f ff ff ff ff ff ff ff ff ff ff ff ff ...O............
70. 00000450 ff ff ff 00 00 00 ff ff 9f ff ff fd f9 fa ff ff ................
71. 00000460 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff ff ff .......o........
72. 00000470 ff ff ff ff ff ff ff 00 00 00 ff ff 1f ff 0f fd ................
73. 00000480 e1 fc 7f ff ff ff ff ff ff ff fe 2f ff ff ff ff .........../....
74. 00000490 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
75. 000004a0 1f fe 7f fb e7 ff 3f ff ff ff ff ff ff ff fe 6f ......?........o
76. 000004b0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
77. 000004c0 00 00 ff ff 5f fe ff fb f3 ff 1f ff ff ff ff ff ....\_...........
78. 000004d0 ff ff fe 0f ff ff ff ff ff ff ff ff ff ff ff ff ................
79. 000004e0 ff ff ff 00 00 00 ff ff 5f fd ff fb e1 ff 1f ff ........\_.......
80. 000004f0 ff ff ff ff ff ff fe a7 ff ff ff ff ff ff ff ff ................
81. 00000500 ff ff ff ff ff ff ff 00 00 00 ff ff df fd ff fb ................
82. 00000510 e1 ff 0f ff ff ff ff ff ff ff fe cf ff ff ff ff ................
83. 00000520 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff fe ................
84. 00000530 0f ff ff f7 e1 fe 07 ff ff ff ff ff ff ff fe 6f ...............o
85. 00000540 ff ff ff ff ff 80 1f ff ff ff ff ff ff ff ff 00 ................
86. 00000550 00 00 ff fe 9f ff ff e7 f5 c0 f7 ff ff ff ff ff ................
87. 00000560 ff ff fe ef ff ff ff ff e0 3d c3 ff ff ff ff ff .........=......
88. 00000570 ff ff ff 00 00 00 ff fe 9f ff ff ef ff ff f7 ff ................
89. 00000580 ff ff ff ff ff ff fe ef ff ff ff ff 40 3f 38 ff ............@?8.
90. 00000590 ff ff ff ff ff ff ff 00 00 00 ff fe 9f ff ff df ................
91. 000005a0 ff ff f7 ff ff ff ff ff ff ff fe 6f ff ff ff ff ...........o....
92. 000005b0 80 bf 9e 1f ff ff ff ff ff ff ff 00 00 00 ff fe ................
93. 000005c0 1f df ff df f7 ff f7 ff ff ff ff ff ff ff fe ef ................
94. 000005d0 ff ff ff ff c2 1f ff ef ff ff ff ff ff ff ff 00 ................
95. 000005e0 00 00 ff fe cf df ff 3f f3 ff ef ff ff ff ff ff .......?........
96. 000005f0 ff ff fe ef ff ff ff ff c1 8f df e7 ff ff ff ff ................
97. 00000600 ff ff ff 00 00 00 ff fe af bf ff ff fb ff ef ff ................
98. 00000610 ff ff ff ff ff ff fe ef ff ff ff ff c3 ee af eb ................
99. 00000620 ff ff ff ff ff ff ff 00 00 00 ff fe af 7f f8 ff ................
100. 00000630 fd ff cf ff ff ff ff ff ff ff fc 6f ff ff ff ff ...........o....
101. 00000640 e0 e2 ff e1 ff ff ff ff ff ff ff 00 00 00 ff fe ................
102. 00000650 ee ff f7 ff f9 ff df ff ff ff ff ff ff ff fe 6f ...............o
103. 00000660 ff ff ff ff f8 ff 1f fc ff ff ff ff ff ff ff 00 ................
104. 00000670 00 00 ff fe ec ff ef ff fb ff bf ff ff ff ff ff ................
105. 00000680 ff ff fe 6f ff ff ff ff fc 3c cf 7e ff ff ff ff ...o.....<.~....
106. 00000690 ff ff ff 00 00 00 ff fe c9 ff df ff f7 ff 7f ff ................
107. 000006a0 ff ff ff ff ff ff fe ef ff ff ff ff fc 20 ef be ............. ..
108. 000006b0 7f ff ff ff ff ff ff 00 00 00 ff fc 83 ff bf ff ................
109. 000006c0 e7 fe 7f ff ff ff ff ff ff ff fe 0f ff ff ff ff ................
110. 000006d0 fe 03 f7 ef 3f ff ff ff ff ff ff 00 00 00 ff fd ....?...........
111. 000006e0 8f ff bf ff ef ff ff ff ff ff ff ff ff ff fe af ................
112. 000006f0 ff ff ff ff ff af f2 ff bf ff ff ff ff ff ff 00 ................
113. 00000700 00 00 ff f9 cf fc 7f ff ef fe ff ff ff ff ff ff ................
114. 00000710 ff ff fe cf ff ff ff ff ff bf f8 77 9f ff ff ff ...........w....
115. 00000720 ff ff ff 00 00 00 ff fb bf fd ff ff 7f fe ff ff ................
116. 00000730 ff ff ff ff ff ff fe ef ff ff ff ff ff ff b8 3f ...............?
117. 00000740 ef ff ff ff ff ff ff 00 00 00 ff f9 7f fb ff fe ................
118. 00000750 ff ff ff ff ff ff ff ff ff ff fe cf ff ff ff ff ................
119. 00000760 ff ff 81 1d e7 ff ff ff ff ff ff 00 00 00 ff f8 ................
120. 00000770 ff f3 ff f8 ff fe ff ff ff ff ff ff ff ff fe ef ................
121. 00000780 ff ff ff ff ff ff 80 0d f7 ff ff ff ff ff ff 00 ................
122. 00000790 00 00 ff fb ff e7 ff e7 ff fd ff ff ff ff ff ff ................
123. 000007a0 ff ff fe ef ff ff ff ff ff ff 80 8f fb ff ff ff ................
124. 000007b0 ff ff ff 00 00 00 ff f7 ff ef ff df ff fb ff ff ................
125. 000007c0 ff ff ff ff ff ff fe cf ff ff ff ff ff ff c3 e6 ................
126. 000007d0 fb ff ff ff ff ff ff 00 00 00 ff f7 ff ef ff 3f ...............?
127. 000007e0 ff fb ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
128. 000007f0 ff ff e3 f7 7b ff ff ff ff ff ff 00 00 00 ff f7 ....{...........
129. 00000800 ff ef f2 7f ff fb ff ff ff ff ff ff ff ff fe ef ................
130. 00000810 ff ff ff ff ff ff f0 f0 fc ff ff ff ff ff ff 00 ................
131. 00000820 00 00 ff f7 ff ef e7 ff ff bf ff ff ff ff ff ff ................
132. 00000830 ff ff ff ef ff ff ff ff ff ff fa f5 5e 7f ff ff ............^...
133. 00000840 ff ff ff 00 00 00 ff e7 ff df bf ff fc ff ff ff ................
134. 00000850 ff ff ff ff ff ff fe ef ff ff ff ff ff ff fc 21 ...............!
135. 00000860 bf 3f ff ff ff ff ff 00 00 00 ff ef ff 5d ff ff .?...........]..
136. 00000870 fd ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
137. 00000880 ff ff fe 38 bf df ff ff ff ff ff 00 00 00 ff ef ...8............
138. 00000890 ff c3 ff ff f3 ff ff ff ff ff ff ff ff ff fe 6f ...............o
139. 000008a0 ff ff ff ff ff ff fe 0f 37 df ff ff ff ff ff 00 ........7.......
140. 000008b0 00 00 ff ef ff 87 bf ff e7 ff ff ff ff ff ff ff ................
141. 000008c0 ff ff ff 2f ff ff ff ff ff ff ff ff eb ef ff ff .../............
142. 000008d0 ff ff ff 00 00 00 ff ef ff 62 7f ff 1f ff ff ff .........b......
143. 000008e0 ff ff ff ff ff ff fe 0f ff ff ff ff ff ff ff ff ................
144. 000008f0 ab e7 ff ff ff ff ff 00 00 00 ff df ff fc 3f fe ..............?.
145. 00000900 7f ff ff ff ff ff ff ff ff ff fe 4f ff ff ff ff ...........O....
146. 00000910 ff ff ff ff c5 f3 ff ff ff ff ff 00 00 00 ff df ................
147. 00000920 ff ff c0 f9 ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
148. 00000930 ff ff ff ff ff ff ff ff f3 fc ff ff ff ff ff 00 ................
149. 00000940 00 00 ff df ff ff ff 6f ff ff ff ff ff ff ff ff .......o........
150. 00000950 ff ff ff 6f ff ff ff ff ff ff ff ff ff ff 7f ff ...o............
151. 00000960 ff ff ff 00 00 00 ff ff ff ff ff 9f ff ff ff ff ................
152. 00000970 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff ff ff .......o........
153. 00000980 f7 ff 7f ff ff ff ff 00 00 00 ff ff ff ff ff e3 ................
154. 00000990 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
155. 000009a0 ff ff ff ff e1 ff 7f ff ff ff ff 00 00 00 ff df ................
156. 000009b0 ff ff ff fc 7f ff ff ff ff ff ff ff ff ff fe 6f ...............o
157. 000009c0 ff ff ff ff ff ff ff ff e1 ff 7f ff ff ff ff 00 ................
158. 000009d0 00 00 ff ef ff ff ff ff 87 ff ff ff ff ff ff ff ................
159. 000009e0 ff ff fe ef ff ff ff ff ff ff ff ff cf f7 7f ff ................
160. 000009f0 ff ff ff 00 00 00 ff f3 ff ff ff ff f8 3f ff ff .............?..
161. 00000a00 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff fe 00 .......o........
162. 00000a10 3f fe 7f ff ff ff ff 00 00 00 ff ed ff ff ff ff ?...............
163. 00000a20 ff bf ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
164. 00000a30 ff ff f3 ff ff fe ff ff ff ff ff 00 00 00 ff ee ................
165. 00000a40 7f ff ff ff ff cf ff ff ff ff ff ff ff ff fe 6f ...............o
166. 00000a50 ff ff ff ff ff ff 9f ff ff fe ff ff ff ff ff 00 ................
167. 00000a60 00 00 ff 9b 9f ff ff ff ff f3 ff ff ff ff ff ff ................
168. 00000a70 ff ff fc 6f ff ff ff ff ff f2 7f ff ff de 7f ff ...o............
169. 00000a80 ff ff ff 00 00 00 ff bb ef ff ff ff ff fb ff ff ................
170. 00000a90 ff ff ff ff ff ff ff 6f ff ff ff ff ff cf ff ff .......o........
171. 00000aa0 ff fd 7f ff ff ff ff 00 00 00 ff bf ff ff ff ff ................
172. 00000ab0 ff fb ff ff ff ff ff ff ff ff fe 2f ff ff ff ff .........../....
173. 00000ac0 ff bf ff ff ff fd bf ff ff ff ff 00 00 00 ff a7 ................
174. 00000ad0 f1 ff ff ff ff f9 ff ff ff ff ff ff ff ff fe 0f ................
175. 00000ae0 ff ff ff ff c0 ff f7 ff ff e3 8f ff ff ff ff 00 ................
176. 00000af0 00 00 ff b7 f6 7f ff ff ff ff ff ff ff ff ff ff ................
177. 00000b00 ff ff fe 6f ff ff ff fe 3f ff ff ff ff d3 ff ff ...o....?.......
178. 00000b10 ff ff ff 00 00 00 ff b7 ff 7f ff ff ff fe ff ff ................
179. 00000b20 ff ff ff ff ff ff fe 6f ff ff ff fc ff ff ff ff .......o........
180. 00000b30 ff 99 ef ff ff ff ff 00 00 00 ff a7 ff 8f ff ff ................
181. 00000b40 ff fe 7f ff ff ff ff ff ff ff fe 6f ff ff ff e1 ...........o....
182. 00000b50 ff ff ff ff fe 0c e7 ff ff ff ff 00 00 00 ff bf ................
183. 00000b60 bf cf ff ff ff ff 1f ff ff ff ff ff ff ff fe 6f ...............o
184. 00000b70 ff ff ff cf ff ff ff ff fd ee f3 ff ff ff ff 00 ................
185. 00000b80 00 00 ff bf f7 e7 ff ff ff ff 8f ff ff ff ff ff ................
186. 00000b90 ff ff ff ef ff ff ff df ff ff ff ff fb f7 7b ff ..............{.
187. 00000ba0 ff ff ff 00 00 00 ff bf d7 e7 ff ff ff ff 87 ff ................
188. 00000bb0 ff ff ff ff ff ff fe ef ff ff ff 1f ff ff ff ff ................
189. 00000bc0 f3 f7 fd ff ff ff ff 00 00 00 ff 9b d3 e1 ff ff ................
190. 00000bd0 ff ff c3 ff ff ff ff ff ff ff fe 6f ff ff f8 7f ...........o....
191. 00000be0 ff ff ff ff 83 f1 7c ff ff ff ff 00 00 00 ff db ......|.........
192. 00000bf0 db fd ff ff ff ff e3 ff ff ff ff ff ff ff fe 6f ...............o
193. 00000c00 ff ff fb ff ff ff ff ff bc 00 7e ff ff ff ff 00 ..........~.....
194. 00000c10 00 00 ff eb db fd ff ff ff ff e4 ff ff ff ff ff ................
195. 00000c20 ff ff fe 6f ff ff e7 ff ff ff ff fc 7f f7 fe ff ...o............
196. 00000c30 ff ff ff 00 00 00 ff ff df fc ff ff ff ff e6 ff ................
197. 00000c40 ff ff ff ff ff ff fe 67 ff ff df ff ff ff ff fd .......g........
198. 00000c50 ff f7 fe ff ff ff ff 00 00 00 ff ef 9f fe 7f ff ................
199. 00000c60 ff ff f7 7f ff ff ff ff ff ff fe 27 ff ff 9f ff ...........'....
200. 00000c70 ff ff ff f7 ff ff ff ff ff ff ff 00 00 00 ff eb ................
201. 00000c80 df fd ff ff ff ff e7 bf ff ff ff ff ff ff fe 6f ...............o
202. 00000c90 ff ff 7f ff ff ff ff ef ff ff fd ff ff ff ff 00 ................
203. 00000ca0 00 00 ff eb df fd ff ff ff ff f1 bf ff ff ff ff ................
204. 00000cb0 ff ff fe 2f ff fe 7f ff ff ff ff 9f ff ff fd ff .../............
205. 00000cc0 ff ff ff 00 00 00 ff eb cb fb ff ff ff ff e0 0f ................
206. 00000cd0 ff ff ff c0 04 0f fe 0f ff fc ff ff ff ff fe 7f ................
207. 00000ce0 ff ff ed ff ff ff ff 00 00 00 ff eb eb f7 ff ff ................
208. 00000cf0 ff ff e8 2f ff ff c0 1f ff e0 1e 6f ff f1 ff ff .../.......o....
209. 00000d00 ff ff fc ff ff ff fd ff ff ff ff 00 00 00 ff e7 ................
210. 00000d10 f3 ef ff ff ff ff f7 03 ff fe 3f ff ff ff c0 2f ..........?..../
211. 00000d20 ff ef ff ff ff ff fb ff ff ff fb ff ff ff ff 00 ................
212. 00000d30 00 00 ff ef f7 ef ff ff ff ff ff 80 ff 00 ff ff ................
213. 00000d40 ff ff f8 0e 3f df ff ff ff ff e7 ff ff ff ef ff ....?...........
214. 00000d50 ff ff ff 00 00 00 ff f7 f7 ff ff ff ff ff ff e0 ................
215. 00000d60 00 0f ff ff ff ff fe 0c 07 df ff ff ff ff 87 ff ................
216. 00000d70 ff ff cf ff ff ff ff 00 00 00 ff d7 f7 ef ff ff ................
217. 00000d80 ff ff ff ff ff ff ff ff ff ff ff e0 00 3f ff ff .............?..
218. 00000d90 ff f8 0f ff bf f0 3f ff ff ff ff 00 00 00 ff d7 ......?.........
219. 00000da0 f7 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fa ................
220. 00000db0 36 ff ff ff ff f9 db ff ff cf ff ff ff ff ff 00 6...............
221. 00000dc0 00 00 ff d7 e3 df ff ff ff ff ff ff ff ff ff ff ................
222. 00000dd0 ff ff ff e0 00 ff ff ff ff f8 07 ff f8 9f ff ff ................
223. 00000de0 ff ff ff 00 00 00 ff ff f3 ff ff ff ff ff ff ff ................
224. 00000df0 ff ff ff ff ff ff ff e0 09 ff ff ff ff ff ff ff ................
225. 00000e00 8f ff ff ff ff ff ff 00 00 00 ff f3 fb df ff ff ................
226. 00000e10 ff ff ff ff ff ff ff ff ff ff ff dc 0d ff ff ff ................
227. 00000e20 ff ff fb fc ff ff ff ff ff ff ff 00 00 00 ff b9 ................
228. 00000e30 fb df ff ff ff ff ff ff ff ff ff ff ff ff ff 8f ................
229. 00000e40 9b ff ff ff ff ff fb 81 ff ff ff ff ff ff ff 00 ................
230. 00000e50 00 00 ff fc f5 ff ff ff ff ff ff ff ff ff ff ff ................
231. 00000e60 ff ff ff c8 db ff ff ff ff ff fa 7f ff ff ff ff ................
232. 00000e70 ff ff ff 00 00 00 ff fe f7 ff ff ff ff ff ff ff ................
233. 00000e80 ff ff ff ff ff ff ff 88 3b ff ff ff ff ff f8 ff ........;.......
234. 00000e90 ff ff ff ff ff ff ff 00 00 00 ff de 6e bf ff ff ............n...
235. 00000ea0 ff ff ff ff ff ff ff ff ff ff ff 80 33 ff ff ff ............3...
236. 00000eb0 ff ff f9 ff ff ff ff ff ff ff ff 00 00 00 ff df ................
237. 00000ec0 af bf ff ff ff ff ff ff ff ff ff ff ff ff ff 94 ................
238. 00000ed0 77 ff ff ff ff ff fb ff ff ff ff ff ff ff ff 00 w...............
239. 00000ee0 00 00 ff df af ff ff ff ff ff ff ff ff ff ff ff ................
240. 00000ef0 ff ff ff 14 7f ff ff ff ff ff fb ff ff ff ff ff ................
241. 00000f00 ff ff ff 00 00 00 ff df ae ff ff ff ff ff ff ff ................
242. 00000f10 ff ff ff ff ff ff ff 28 7f ff ff ff ff ff f9 ff .......(........
243. 00000f20 ff ff ff ff ff ff ff 00 00 00 ff ff ef ff ff ff ................
244. 00000f30 ff ff ff ff ff ff ff ff ff ff ff 7e ff ff ff ff ...........~....
245. 00000f40 ff ff fd ff ff ff ff ff ff ff ff 00 00 00 ff ef ................
246. 00000f50 6d ff ff ff ff ff ff ff ff ff ff ff ff ff fe fd m...............
247. 00000f60 ff ff ff ff ff ff fd ff ff ff ff ff ff ff ff 00 ................
248. 00000f70 00 00 ff ef 6d ff ff ff ff ff ff ff ff ff ff ff ....m...........
249. 00000f80 ff ff fe fd ff ff ff ff ff ff fd ff ff ff ff ff ................
250. 00000f90 ff ff ff 00 00 00 ff f2 ed ff ff ff ff c7 ff ff ................
251. 00000fa0 ff ff ff ff ff ff fd fd ff ff ff ff ff ff fc ff ................
252. 00000fb0 ff ff ff ff ff ff ff 00 00 00 ff fb fd ff ff ff ................
253. 00000fc0 ff c3 ff ff ff ff ff ff ff ff fd fd ff ff ff ff ................
254. 00000fd0 ff ff fc ff ff ff ff ff ff ff ff 00 00 00 ff f5 ................
255. 00000fe0 df ff ff ff ff e8 ff 9f ff ff ff ff ff ff fb fb ................
256. 00000ff0 ff ff ff ff ff ff fe ff ff ff ff ff ff ff ff 00 ................
257. 00001000 00 00 ff fe df bf ff ff ff ea 0f 8f ............
258. 22 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x22
259. class = TS\_SECONDARY (0x02)
260. orderType = TS\_ALTSEC\_GDIP\_CACHE\_FIRST (0x08)
261. 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::Flags = 0x00
262. 04 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::CacheType = GDIP\_CACHE\_OBJECT\_IMAGE (0x0004)
263. 00 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::CacheIndex = 0x0000
264. 00 10 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::cbSize = 0x1000 = 4096 bytes
265. 1c 22 00 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::cbTotalSize = 0x0000221c = 8732 bytes
266. The next 4,096 bytes are the EMF+ records (DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::emfRecords). These
267. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

#### Example 2

The following is the second example of an annotated dump of a [Draw GDI+ Cache First (section 2.2.2.2.1.3.6.2)](#Section_53dbc9c54e3f4b17a256baf43c582146) Alternate Secondary Drawing Order.

1. 00000000 22 00 01 00 08 00 28 00 28 00 00 00 00 00 00 00 ".....(.(.......
2. 00000010 00 00 00 01 00 00 00 00 04 00 03 03 00 00 80 3f ...............?
3. 00000020 00 00 00 00 00 00 00 00 00 00 80 3f 00 00 20 43 ...........?.. C
4. 00000030 00 00 00 00 ....
5. 22 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x22
6. class = TS\_SECONDARY (0x02)
7. orderType = TS\_ALTSEC\_GDIP\_CACHE\_FIRST (0x08)
8. 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::Flags = 0x00
9. 01 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::CacheType = GDIP\_CACHE\_GRAPHICS\_DATA (0x0001)
10. 08 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::CacheIndex = 0x0008
11. 28 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::cbSize = 0x0028 = 40 bytes
12. 28 00 00 00 -> DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::cbTotalSize = 0x00000028 = 40 bytes
13. The next 40 bytes are the EMF+ records (DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER::emfRecords). These
14. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

### Draw GDI+ Cache Next

The following is an annotated dump of a [Draw GDI+ Cache Next (section 2.2.2.2.1.3.6.3)](#Section_20da0b5e8cc94805998e03e9f216bade) Alternate Secondary Drawing Order.

1. 00000000 26 00 04 00 00 00 00 10 ff ff ff ff ff ff fb fb &...............
2. 00000010 ff ff ff ff ff ff fe ff ff ff ff ff ff ff ff 00 ................
3. 00000020 00 00 ff f6 df bf ff ff ff eb e3 af ff ff ff ff ................
4. 00000030 ff ff f9 f3 ff ff ff ff ff ff c2 ff ff ff ff ff ................
5. 00000040 ff ff ff 00 00 00 ff f7 5e bf ff ff ff eb f0 2f ........^....../
6. 00000050 ff ff ff ff ff ff fb f7 ff ff ff ff ff ff 18 ff ................
7. 00000060 ff ff ff ff ff ff ff 00 00 00 ff fb 3e ff ff ff ............>...
8. 00000070 ff e9 fc 23 ff ff ff ff ff ff fb f7 ff ff ff ff ...#............
9. 00000080 ff 81 d8 ff ff ff ff ff ff ff ff 00 00 00 ff fd ................
10. 00000090 ac df ff ff ff e6 00 00 7f ff 0f ff ff ff f0 67 ...............g
11. 000000a0 1f ff ff ff f8 3e 02 ff ff ff ff ff ff ff ff 00 .....>..........
12. 000000b0 00 00 ff fe fe ff ff ff ff e3 3f 2e 1f ff 60 00 ..........?...`.
13. 000000c0 00 00 00 00 5f ff ff ff 82 80 01 ff ff ff ff ff ....\_...........
14. 000000d0 ff ff ff 00 00 00 ff ff df 7f ff ff ff 80 1f 47 ...............G
15. 000000e0 cf ff 7f ff ff ff ff ff 5f ff ff fe 61 0f fc ff ........\_...a...
16. 000000f0 ff ff ff ff ff ff ff 00 00 00 ff fe ff 6f ff ff .............o..
17. 00000100 ff 99 cb e3 e7 ff 7f fd ff ff ff fb 9f ff ff f8 ................
18. 00000110 fd 3f fd ff ff ff ff ff ff ff ff 00 00 00 ff ff .?..............
19. 00000120 ff 0f ff ff ff c9 e6 e9 f9 ff 3f ff 7f ff d5 7f ..........?.....
20. 00000130 bf ff ff e6 b8 7f fd ff ff ff ff ff ff ff ff 00 ................
21. 00000140 00 00 ff ff 2f cf ff ff ff e1 80 e0 58 1f 38 80 ..../.......X.8.
22. 00000150 00 00 00 03 bf ff fe 07 03 ff f9 ff ff ff ff ff ................
23. 00000160 ff ff ff 00 00 00 ff ff bf ff ff ff ff f9 e8 60 ...............`
24. 00000170 14 c7 39 f8 7f ff fc 83 bf ff fc c2 3f ff fb ff ..9.........?...
25. 00000180 ff ff ff ff ff ff ff 00 00 00 ff ff cf df ff ff ................
26. 00000190 ff f9 c3 87 c0 01 39 00 01 ff ff f3 3f ff 80 c0 ......9.....?...
27. 000001a0 8f ff fb ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
28. 000001b0 ef df ff ff ff fc e3 ee 1b 74 10 3a 00 3f ff f7 .........t.:.?..
29. 000001c0 7f ff 2f 1c 03 ff f7 ff ff ff ff ff ff ff ff 00 ../.............
30. 000001d0 00 00 ff ff f7 c7 ff ff ff fe e1 c0 4b 70 1c 01 ............Kp..
31. 000001e0 00 0f ff f7 7f fc 78 30 13 ff e7 ff ff ff ff ff ......x0........
32. 000001f0 ff ff ff 00 00 00 ff ff fa 07 ff ff ff fe 0f 83 ................
33. 00000200 c3 78 1c 00 00 03 ff f7 7f f9 f9 84 18 4f e7 ff .x...........O..
34. 00000210 ff ff ff ff ff ff ff 00 00 00 ff ff ff f3 ff ff ................
35. 00000220 ff ff f7 3f e0 60 00 11 00 00 ff f7 7f e3 f2 14 ...?.`..........
36. 00000230 18 0f ef ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
37. 00000240 ff ff ff ff ff ff f2 7f e3 e0 00 00 00 00 ff f7 ................
38. 00000250 7f cf e4 12 09 e7 ef ff ff ff ff ff ff ff ff 00 ................
39. 00000260 00 00 ff ff ff fd ff ff ff ff ff ff e3 c0 00 00 ................
40. 00000270 00 00 1f f7 7e 1c 00 60 00 27 df ff ff ff ff ff ....~..`.'......
41. 00000280 ff ff ff 00 00 00 ff ff ff fe ff ff ff ff ff ff ................
42. 00000290 f0 04 01 ea 60 01 df f7 7c f3 f7 ea 1d 3f ff ff ....`...|....?..
43. 000002a0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff 7f ff ................
44. 000002b0 ff ff ff ff f0 0c 00 00 01 e7 cf f7 73 e7 e3 c6 ............s...
45. 000002c0 0d 7f bf ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
46. 000002d0 ff ff bf ff ff ff ff ff f0 08 0c 41 41 83 cf f7 ...........AA...
47. 000002e0 07 9f f0 13 98 ff bf ff ff ff ff ff ff ff ff 00 ................
48. 000002f0 00 00 ff ff ff ff bf ff ff ff ff ff fe 10 7b ff ..............{.
49. 00000300 f7 ff ef f7 7e 7f fc 09 3b ff bf ff ff ff ff ff ....~...;.......
50. 00000310 ff ff ff 00 00 00 ff ff ff ff bf ff ff ff ff ff ................
51. 00000320 ff 9e 7f ff ff fe c7 f7 7e ff ff 00 03 ff bf ff ........~.......
52. 00000330 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff df ff ................
53. 00000340 ff ff ff ff ff 97 ff ff bf ff c7 f7 00 ff ff 8e ................
54. 00000350 5f fe 3f ff ff ff ff ff ff ff ff 00 00 00 ff ff \_.?.............
55. 00000360 ff ff e7 ff ff ff ff ff ff cf 7f ff ff fe 57 f6 ..............W.
56. 00000370 07 ff ff e0 00 00 3f ff ff ff ff ff ff ff ff 00 ......?.........
57. 00000380 00 00 ff ff ff ff fb ff ff ff ff ff fe ce ff ff ................
58. 00000390 ff ff f7 f6 ff ff ff 80 ff ff 3f ff ff ff ff ff ..........?.....
59. 000003a0 ff ff ff 00 00 00 ff ff ff ff ff 7f ff ff ff ff ................
60. 000003b0 ff ce ff ff ff ff f7 f6 ff ff c0 02 ff ff 3f ff ..............?.
61. 000003c0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff 7f ................
62. 000003d0 ff ff ff ff ff cf ff ff ff ff f7 f6 ff fc 3f fe ..............?.
63. 000003e0 d0 3c 3f ff ff ff ff ff ff ff ff 00 00 00 ff ff .<?.............
64. 000003f0 ff ff ff bf ff ff ff ff ff cf ff ff ff ff f7 f6 ................
65. 00000400 ff e1 ff 54 05 ff 3f ff ff ff ff ff ff ff ff 00 ...T..?.........
66. 00000410 00 00 ff ff ff ff ff df ff ff ff ff ff df ff ff ................
67. 00000420 ff ff e7 f6 7e 0f ff c0 ff ff 7f ff ff ff ff ff ....~...........
68. 00000430 ff ff ff 00 00 00 ff ff ff ff ff e3 ff ff ff ff ................
69. 00000440 ff 0f ff ff ff ff f7 f4 80 ff c0 0f ff e7 7f ff ................
70. 00000450 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff fc ................
71. 00000460 ff ff ff ff f8 3f ff ff ff ff c7 e4 7f c0 7c 7f .....?........|.
72. 00000470 ff f5 7f ff ff c0 3f ff ff ff ff 00 00 00 ff ff ......?.........
73. 00000480 ff ff ff fe 3f ff ff ff c3 7f ff ff ff ff e7 ec ....?...........
74. 00000490 60 0f fd 3f ff fd 7f ff fe 03 cf ff ff ff ff 00 `..?............
75. 000004a0 00 00 ff ff ff ff ff ff cf ff ff fc 1e ff ff ff ................
76. 000004b0 ff ff c7 e4 03 ff f9 bf ff fe 7f ff f8 7f ef ff ................
77. 000004c0 ff ff ff 00 00 00 ff ff ff ff ff ff e3 ff ff e0 ................
78. 000004d0 c9 ff ff ff ff ff e7 ec 3f ff f9 bf fd ff 7f ff ........?.......
79. 000004e0 fb df ef ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
80. 000004f0 fd ff ff 81 c7 ff fd fb 7f b0 a7 e4 7f ff f1 bf ................
81. 00000500 ff ff 7f ff f3 97 ff ff ff ff ff 00 00 00 ff ff ................
82. 00000510 ff ff ff ff ff 00 40 0c 1f e9 bf 5a 06 80 03 ee ......@....Z....
83. 00000520 ff ff 0f 7f ff ff 7f 0f c3 87 ef ff ff ff ff 00 ................
84. 00000530 00 00 ff ff ff ff ff ff ff ff 00 00 fb 00 40 00 ..............@.
85. 00000540 40 0c 03 ee ff 00 fc ff fb ff 7f 21 d7 c7 ef ff @..........!....
86. 00000550 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff 80 ff ................
87. 00000560 ff 20 00 02 00 04 03 e6 70 3f f9 ff ff ff 7f 48 . ......p?.....H
88. 00000570 d7 ff ef ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
89. 00000580 ff ff 8f e1 d0 00 00 00 40 24 0b e3 77 ff 73 ff ........@$..w.s.
90. 00000590 ff ff 3f 78 e7 ff f7 ff ff ff ff 00 00 00 ff ff ..?x............
91. 000005a0 ff ff ff ff ff ff 15 8b 40 00 00 00 00 00 9b e9 ........@.......
92. 000005b0 47 fe 73 ff fb ff 3e 6d 4f ff fb ff ff ff ff 00 G.s...>mO.......
93. 000005c0 00 00 ff ff ff ff ff ff ff ff 00 04 00 00 00 00 ................
94. 000005d0 00 00 13 b4 0f fc c7 ff fc ff 3e 2c 1f ff f9 ff ..........>,....
95. 000005e0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff 20 00 .............. .
96. 000005f0 00 7f ff ff f8 00 13 1b bf f1 8f ff e7 ff be 70 ...............p
97. 00000600 1f ff ed ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
98. 00000610 ff ff a0 20 1f ff ff ff fe 00 10 00 3f c7 3f ff ... ........?.?.
99. 00000620 c7 ff be 3c 9f ff ef ff ff ff ff 00 00 00 ff ff ...<............
100. 00000630 ff ff ff ff ff ff 80 00 ff ff ff ff ff 00 06 0e ................
101. 00000640 3e 1e 3f ff c1 ff be 1c 07 ff fc ff ff ff ff 00 >.?.............
102. 00000650 00 00 ff ff ff ff ff ff ff ff e0 47 ff ff ff ff ...........G....
103. 00000660 ff e3 ee 6f 7e 7c ff ff af ff bf 8d d7 fe fe ff ...o~|..........
104. 00000670 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff e8 0f ................
105. 00000680 ff ff ff ff ff f0 5e 6f 7e fc ff ff af ff bf cd ......^o~.......
106. 00000690 d7 fe fe ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
107. 000006a0 ff ff f3 8f ff ff ff ff ff fc 3e 6f 78 fc ff fe ..........>ox...
108. 000006b0 27 ff bf e4 d7 fc ff 7f ff ff ff 00 00 00 ff ff '...............
109. 000006c0 ff ff ff ff ff ff fc 3f ff ff ff ff ff ff fe 6f .......?.......o
110. 000006d0 3b fc ff fe f7 ff bf f5 f7 fd ff 7f ff ff ff 00 ;...............
111. 000006e0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
112. 000006f0 ff ff fe 6f 33 fc 3f f8 e7 ff bf f5 f7 fd fd 7f ...o3.?.........
113. 00000700 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
114. 00000710 ff ff ff ff ff ff fe 6f 87 fc 03 e3 f7 ff 9f f4 .......o........
115. 00000720 37 fb ff ff ff ff ff 00 00 00 ff ff ff ff ff ff 7...............
116. 00000730 ff ff ff ff ff ff ff ff ff ff fe 6f 87 81 f0 07 ...........o....
117. 00000740 fb ff df f8 a7 ff fd bf ff ff ff 00 00 00 ff ff ................
118. 00000750 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
119. 00000760 f7 3f ff ff fb ff cf f8 b7 ff ff bf ff ff ff 00 .?..............
120. 00000770 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
121. 00000780 ff ff fe 2f f6 7f ff ff f9 ff ef f9 37 ff ff bf .../........7...
122. 00000790 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
123. 000007a0 ff ff ff ff ff ff fe 2f f4 ff ff ff fd ff e7 fd ......./........
124. 000007b0 27 ff ff bf ff ff ff 00 00 00 ff ff ff ff ff ff '...............
125. 000007c0 ff ff ff ff ff ff ff ff ff ff fe 2f f0 ff ff ff .........../....
126. 000007d0 fd ff f7 fd 37 ff ff bf ff ff ff 00 00 00 ff ff ....7...........
127. 000007e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 0f ................
128. 000007f0 f1 ff fd 7f fd ff f3 fd bf ff ff df ff ff ff 00 ................
129. 00000800 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
130. 00000810 ff ff fe 0f f3 ff f9 ff fd ff f3 fd b3 fb ff df ................
131. 00000820 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
132. 00000830 ff ff ff ff ff ff fe 6f f7 ff f9 ff f9 ff fb fc .......o........
133. 00000840 93 ff ff df ff ff ff 00 00 00 ff ff ff ff ff ff ................
134. 00000850 ff ff ff ff ff ff ff ff ff ff fe 4f f7 ff f9 ff ...........O....
135. 00000860 fb ff fb fd 13 ff fb a7 ff ff ff 00 00 00 ff ff ................
136. 00000870 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
137. 00000880 f7 ff f9 ff fb ff f9 fd a7 fb fb 91 ff ff ff 00 ................
138. 00000890 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
139. 000008a0 ff ff fe 6f f3 ff f9 ff f7 ff f9 fd ad fd fd 81 ...o............
140. 000008b0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
141. 000008c0 ff ff ff ff ff ff ff 6f fb ff f9 ff e7 ff fc fd .......o........
142. 000008d0 ab fd ff a3 ff ff ff 00 00 00 ff ff ff ff ff ff ................
143. 000008e0 ff ff ff ff ff ff ff ff ff ff fe 4f f9 ff f9 ff ...........O....
144. 000008f0 cf ff fe f8 01 ff f9 d0 ff ff ff 00 00 00 ff ff ................
145. 00000900 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
146. 00000910 fc ff c1 ff df ff fe e1 9e ff 71 14 7f ff ff 00 ..........q.....
147. 00000920 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
148. 00000930 ff ff ff 2f fe fc 07 fb 9f c3 ff 07 c4 ff e1 50 .../...........P
149. 00000940 3f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ?...............
150. 00000950 ff ff ff ff ff ff fe 0f fe dd ff ff 1f 9b fb 1f ................
151. 00000960 f5 7f e0 58 bf ff ff 00 00 00 ff ff ff ff ff ff ...X............
152. 00000970 ff ff ff ff ff ff ff ff ff ff fe 2f fe fd ff ff .........../....
153. 00000980 00 33 fd 0f f1 bf e0 9c 3f ff ff 00 00 00 ff ff .3......?.......
154. 00000990 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 4f ...............O
155. 000009a0 fe 33 ff f0 00 e7 ff 0f f0 9f e0 0c 9f ff ff 00 .3..............
156. 000009b0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
157. 000009c0 ff ff ff 4f ff 87 ff e1 ff ef ff 0f f9 5f f8 46 ...O.........\_.F
158. 000009d0 9f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
159. 000009e0 ff ff ff ff ff ff fe 4f ff cf ff c7 ff ef ff 07 .......O........
160. 000009f0 f9 5f f8 e6 cf ff ff 00 00 00 ff ff ff ff ff ff .\_..............
161. 00000a00 ff ff ff ff ff ff ff ff ff ff fe 4f ff cf fe 0f ...........O....
162. 00000a10 ff cf 86 09 f8 0f f9 f0 c7 ff ff 00 00 00 ff ff ................
163. 00000a20 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
164. 00000a30 ff df fc f0 df cf b6 09 fd 17 f9 fc b7 ff ff 00 ................
165. 00000a40 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
166. 00000a50 ff ff ff 6f ff df fd e0 3f af b7 0e fc 97 ff fd ...o....?.......
167. 00000a60 bb ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
168. 00000a70 ff ff ff ff ff ff fe 6f ff df f9 e6 03 ef 27 0f .......o......'.
169. 00000a80 3e 03 ff f9 bb ff ff 00 00 00 ff ff ff ff ff ff >...............
170. 00000a90 ff ff ff ff ff ff ff ff ff ff ff 2f ff df fb ef .........../....
171. 00000aa0 cf ef 07 47 9f 83 ff f3 bb ff ff 00 00 00 ff ff ...G............
172. 00000ab0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 2f .............../
173. 00000ac0 ff df fb ef ff e7 4f 87 ef 81 ff f3 99 ff ff 00 ......O.........
174. 00000ad0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
175. 00000ae0 ff ff fe 2f ff cf fb ee ff e2 47 a7 f3 c0 3f f7 .../......G...?.
176. 00000af0 99 ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
177. 00000b00 ff ff ff ff ff ff fe 0f ff cf f3 cf 1f e0 00 c1 ................
178. 00000b10 f8 80 00 03 c8 ff ff 00 00 00 ff ff ff ff ff ff ................
179. 00000b20 ff ff ff ff ff ff ff ff ff ff ff 6f ff cf f7 ef ...........o....
180. 00000b30 9f f8 18 70 ff d6 21 79 e8 ff ff 00 00 00 ff ff ...p..!y........
181. 00000b40 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
182. 00000b50 ff ef f7 8f 9f c0 7e fd 1f d7 07 fc c8 ff ff 00 ......~.........
183. 00000b60 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
184. 00000b70 ff ff fe 6f ff ef f7 df 3f 9e 7e fe 71 97 0f f8 ...o....?.~.q...
185. 00000b80 4b ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff K...............
186. 00000b90 ff ff ff ff ff ff ff 6f ff f7 f7 fe 3f be fe ff .......o....?...
187. 00000ba0 3c 57 28 1e 3b ff ff 00 00 00 ff ff ff ff ff ff <W(.;...........
188. 00000bb0 ff ff ff ff ff ff ff ff ff ff ff 6f ff f7 f7 fe ...........o....
189. 00000bc0 ff 3e fe ff c7 c0 61 c0 13 ff ff 00 00 00 ff ff .>....a.........
190. 00000bd0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
191. 00000be0 ff f9 f7 f0 fc 7e fc fe 10 03 1f ff 07 ff ff 00 .....~..........
192. 00000bf0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
193. 00000c00 ff ff fe 6f ff f9 f7 e7 f9 ee fe e4 0f f0 0f ff ...o............
194. 00000c10 5f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff \_...............
195. 00000c20 ff ff ff ff ff ff ff 2f ff fd f7 c7 fb ee fe 01 ......./........
196. 00000c30 e7 f0 8f fe df ff ff 00 00 00 ff ff ff ff ff ff ................
197. 00000c40 ff ff ff ff ff ff ff ff ff ff ff 2f ff fc f7 cf .........../....
198. 00000c50 fb ee 7c 1d f7 fa cf fe 0f ff ff 00 00 00 ff ff ..|.............
199. 00000c60 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 2f .............../
200. 00000c70 ff fe f3 cf f3 ce fc 7f f7 f9 ef df 2f ff ff 00 ............/...
201. 00000c80 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
202. 00000c90 ff ff fe 2f ff ff 7b cf f7 8e 78 ff b7 fc 63 c3 .../..{...x...c.
203. 00000ca0 af ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
204. 00000cb0 ff ff ff ff ff ff fe 4f ff ff bb cf f7 9f 79 ff .......O......y.
205. 00000cc0 f7 fe e1 c0 af ff ff 00 00 00 ff ff ff ff ff ff ................
206. 00000cd0 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff c1 cf ...........o....
207. 00000ce0 e7 9f 7b f7 f7 ff 00 c0 97 ff ff 00 00 00 ff ff ..{.............
208. 00000cf0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
209. 00000d00 ff ff f9 e7 e7 9f 73 fb f3 ff 08 32 57 ff ff 00 ......s....2W...
210. 00000d10 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
211. 00000d20 ff ff ff 6f ff ff fd e7 f7 9f 77 fb fb ff cc 34 ...o......w....4
212. 00000d30 5f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff \_...............
213. 00000d40 ff ff ff ff ff ff ff 6f ff ff fc e7 f7 9f 77 fb .......o......w.
214. 00000d50 fb ff ec 17 1b ff ff 00 00 00 ff ff ff ff ff ff ................
215. 00000d60 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff fe 20 ...........o...
216. 00000d70 e7 9e e7 fb f9 ff e0 0b 1b ff ff 00 00 00 ff ff ................
217. 00000d80 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
218. 00000d90 ff ff ff 98 c7 9f e7 fb fc ff f8 27 9f ff ff 00 ...........'....
219. 00000da0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
220. 00000db0 ff ff ff 6f ff ff ff dc 13 9f f7 fb fe 7f f8 77 ...o...........w
221. 00000dc0 9b ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
222. 00000dd0 ff ff ff ff ff ff ff 2f ff ff ff e0 03 8f f7 f9 ......./........
223. 00000de0 ff 80 00 72 03 ff ff 00 00 00 ff ff ff ff ff ff ...r............
224. 00000df0 ff ff ff ff ff ff ff ff ff ff ff 2f ff ff ff fe .........../....
225. 00000e00 0b ef f3 fd ff f0 1c fd 1f ff ff 00 00 00 ff ff ................
226. 00000e10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 0f ................
227. 00000e20 ff ff ff ff fa e7 e3 fd ff ff fd fe df ff ff 00 ................
228. 00000e30 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
229. 00000e40 ff ff ff 0f ff ff ff ff f9 e3 fb fc ff ff d9 ff ................
230. 00000e50 1f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
231. 00000e60 ff ff ff ff ff ff ff 4f ff ff ff ff fc f9 fb fe .......O........
232. 00000e70 7f ff f3 ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
233. 00000e80 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
234. 00000e90 fe fc f9 ff 3f ff e7 ff ff ff ff 00 00 00 ff ff ....?...........
235. 00000ea0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
236. 00000eb0 ff ff ff ff fe 7e f9 ff 1f ff ef ff ff ff ff 00 .....~..........
237. 00000ec0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
238. 00000ed0 ff ff ff 6f ff ff ff ff ff 3e 3c ff 80 00 0f ff ...o.....><.....
239. 00000ee0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
240. 00000ef0 ff ff ff ff ff ff ff 6f ff ff ff ff ff bf be ff .......o........
241. 00000f00 ff f3 ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
242. 00000f10 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
243. 00000f20 ff 8f 82 3f ff cf ff ff ff ff ff 00 00 00 ff ff ...?............
244. 00000f30 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
245. 00000f40 ff ff ff ff ff e7 fb 8f ff df ff ff ff ff ff 00 ................
246. 00000f50 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
247. 00000f60 ff ff ff 6f ff ff ff ff ff f0 f8 e7 ff 9f ff ff ...o............
248. 00000f70 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
249. 00000f80 ff ff ff ff ff ff ff 6f ff ff ff ff ff fc 7e e1 .......o......~.
250. 00000f90 ff 9f ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
251. 00000fa0 ff ff ff ff ff ff ff ff ff ff ff 2f ff ff ff ff .........../....
252. 00000fb0 ff ff 02 7b fc 3f ff ff ff ff ff 00 00 00 ff ff ...{.?..........
253. 00000fc0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 0f ................
254. 00000fd0 ff ff ff ff ff ff fc 1f f8 ff ff ff ff ff ff 00 ................
255. 00000fe0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
256. 00000ff0 ff ff ff 0f ff ff ff ff ff ff ff c0 03 ff ff ff ................
257. 00001000 ff ff ff 00 00 00 ff ff ........
258. 26 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x26
259. class = TS\_SECONDARY (0x02)
260. orderType = TS\_ALTSEC\_GDIP\_CACHE\_NEXT (0x09)
261. 00 -> DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER::Flags = 0x00
262. 04 00 -> DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER::CacheType = GDIP\_CACHE\_OBJECT\_IMAGE (0x0004)
263. 00 00 -> DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER::CacheIndex = 0x0000
264. 00 10 -> DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER::cbSize = 0x1000 = 4096 bytes
265. The next 4096 bytes are the EMF+ Records (DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER::emfRecords). These
266. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

### Draw GDI+ Cache End

The following is an annotated dump of a [Draw GDI+ Cache End (section 2.2.2.2.1.3.6.4)](#Section_b44bd2b06c00454db182dbb841131c7a) Alternate Secondary Drawing Order.

1. 00000000 2a 00 04 00 00 00 1c 02 1c 22 00 00 ff ff ff ff \*........"......
2. 00000010 ff ff ff ff ff ff ff ff ff ff ff 4f ff ff ff ff ...........O....
3. 00000020 ff ff ff fc 3f ff ff ff ff ff ff 00 00 00 ff ff ....?...........
4. 00000030 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
5. 00000040 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
6. 00000050 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
7. 00000060 ff ff ff 6f ff ff ff ff ff ff ff ff ff ff ff ff ...o............
8. 00000070 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
9. 00000080 ff ff ff ff ff ff ff 6f ff ff ff ff ff ff ff ff .......o........
10. 00000090 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
11. 000000a0 ff ff ff ff ff ff ff ff ff ff fe 6f ff ff ff ff ...........o....
12. 000000b0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
13. 000000c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 2f .............../
14. 000000d0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
15. 000000e0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
16. 000000f0 ff ff ff 6f ff ff ff ff ff ff ff ff ff ff ff ff ...o............
17. 00000100 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
18. 00000110 ff ff ff ff ff ff ff 2f ff ff ff ff ff ff ff ff ......./........
19. 00000120 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
20. 00000130 ff ff ff ff ff ff ff ff ff ff fe 2f ff ff ff ff .........../....
21. 00000140 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
22. 00000150 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 2f .............../
23. 00000160 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
24. 00000170 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
25. 00000180 ff ff fe 0f ff ff ff ff ff ff ff ff ff ff ff ff ................
26. 00000190 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
27. 000001a0 ff ff ff ff ff ff fe 0f ff ff ff ff ff ff ff ff ................
28. 000001b0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
29. 000001c0 ff ff ff ff ff ff ff ff ff ff fe 4f ff ff ff ff ...........O....
30. 000001d0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
31. 000001e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
32. 000001f0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
33. 00000200 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
34. 00000210 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
35. 00000220 ff ff ff 00 00 00 00 00 ........
36. 2a -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x2a
37. class = TS\_SECONDARY (0x02)
38. orderType = TS\_ALTSEC\_GDIP\_CACHE\_END (0x0a)
39. 00 -> DRAW\_GDIPLUS\_CACHE\_END\_ORDER::Flags = 0x00
40. 04 00 -> DRAW\_GDIPLUS\_CACHE\_END\_ORDER::CacheType = GDIP\_CACHE\_OBJECT\_IMAGE (0x0004)
41. 00 00 -> DRAW\_GDIPLUS\_CACHE\_END\_ORDER::CacheIndex = 0x0000
42. 1c 02 -> DRAW\_GDIPLUS\_CACHE\_END\_ORDER::cbSize = 0x021c = 540 bytes
43. 1c 22 00 00 -> DRAW\_GDIPLUS\_CACHE\_END\_ORDER::cbTotalSize = 0x0000221c = 8732 bytes
44. The next 540 bytes are the EMF+ Records (DRAW\_GDIPLUS\_CACHE\_END\_ORDER::emfRecords). These
45. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

### Draw GDI+ First

The following is an annotated dump of a [Draw GDI+ First (section 2.2.2.2.1.3.6.5)](#Section_03941c5e98d64f61ac74ddb307f1f317) Alternate Secondary Drawing Order.

1. 00000000 16 24 00 10 24 22 00 00 24 22 00 00 08 40 1d 05 .$..$"..$"...@..
2. 00000010 24 22 00 00 02 10 c0 db 01 00 00 00 00 00 00 00 $"..............
3. 00000020 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 ................
4. 00000030 42 4d fe 21 00 00 00 00 00 00 3e 00 00 00 28 00 BM.!......>...(.
5. 00000040 00 00 08 01 00 00 f0 00 00 00 01 00 01 00 00 00 ................
6. 00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
7. 00000060 00 00 00 00 00 00 00 00 00 00 ff ff ff 00 ff ff ................
8. 00000070 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
9. 00000080 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
10. 00000090 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
11. 000000a0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
12. 000000b0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
13. 000000c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
14. 000000d0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
15. 000000e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
16. 000000f0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
17. 00000100 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
18. 00000110 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
19. 00000120 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
20. 00000130 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
21. 00000140 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
22. 00000150 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
23. 00000160 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff d1 ff ................
24. 00000170 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
25. 00000180 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
26. 00000190 ff ff 80 7f ff ff ff ff ff ff ff ff ff ff ff ff ................
27. 000001a0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
28. 000001b0 00 00 ff ff ff ff 7e 7f ff ff ff ff ff ff ff ff ......~.........
29. 000001c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
30. 000001d0 ff ff ff 00 00 00 ff ff ff fe fe 3f ff ff ff ff ...........?....
31. 000001e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
32. 000001f0 ff ff ff ff ff ff ff 00 00 00 ff ff ff fd fe 1f ................
33. 00000200 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
34. 00000210 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
35. 00000220 ff e3 ff 1f ff ff ff ff ff ff ff ff ff ff ff ff ................
36. 00000230 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
37. 00000240 00 00 ff ff ff e5 ff 8f ff ff ff ff ff ff ff ff ................
38. 00000250 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
39. 00000260 ff ff ff 00 00 00 ff ff ff c3 ff c7 ff ff ff ff ................
40. 00000270 ff ff ff ff ff ff fc 2f ff ff ff ff ff ff ff ff ......./........
41. 00000280 ff ff ff ff ff ff ff 00 00 00 ff ff ff c7 ff f3 ................
42. 00000290 ff ff ff ff ff ff ff ff ff ff fd 0f ff ff ff ff ................
43. 000002a0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
44. 000002b0 ff c7 fe f9 ff ff ff ff ff ff ff ff ff ff fe 9f ................
45. 000002c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
46. 000002d0 00 00 ff ff ff c7 f9 fd ff ff ff ff ff ff ff ff ................
47. 000002e0 ff ff fe cf ff ff ff ff ff ff ff ff ff ff ff ff ................
48. 000002f0 ff ff ff 00 00 00 ff ff ff c1 f7 fd ff ff ff ff ................
49. 00000300 ff ff ff ff ff ff fe 8f ff ff ff ff ff ff ff ff ................
50. 00000310 ff ff ff ff ff ff ff 00 00 00 ff ff ff fd cf ff ................
51. 00000320 ff ff ff ff ff ff ff ff ff ff fe c7 ff ff ff ff ................
52. 00000330 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
53. 00000340 ff e0 1f fe ff ff ff ff ff ff ff ff ff ff fe ef ................
54. 00000350 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
55. 00000360 00 00 ff ff ff f8 ff fe ff ff ff ff ff ff ff ff ................
56. 00000370 ff ff fe ef ff ff ff ff ff ff ff ff ff ff ff ff ................
57. 00000380 ff ff ff 00 00 00 ff ff ff fd ff fe ff ff ff ff ................
58. 00000390 ff ff ff ff ff ff fe ef ff ff ff ff ff ff ff ff ................
59. 000003a0 ff ff ff ff ff ff ff 00 00 00 ff ff ff fc ff de ................
60. 000003b0 ff fb ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
61. 000003c0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
62. 000003d0 ff fe ff fe ff e3 ff ff ff ff ff ff ff ff fe ef ................
63. 000003e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
64. 000003f0 00 00 ff ff f3 ff ff fd ff c3 ff ff ff ff ff ff ................
65. 00000400 ff ff fe ef ff ff ff ff ff ff ff ff ff ff ff ff ................
66. 00000410 ff ff ff 00 00 00 ff ff ef ff bf fd ff 8b ff ff ................
67. 00000420 ff ff ff ff ff ff fe ef ff ff ff ff ff ff ff ff ................
68. 00000430 ff ff ff ff ff ff ff 00 00 00 ff ff cf ff bf fd ................
69. 00000440 fc 38 ff ff ff ff ff ff ff ff fe 4f ff ff ff ff .8.........O....
70. 00000450 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
71. 00000460 9f ff ff fd f9 fa ff ff ff ff ff ff ff ff fe 6f ...............o
72. 00000470 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
73. 00000480 00 00 ff ff 1f ff 0f fd e1 fc 7f ff ff ff ff ff ................
74. 00000490 ff ff fe 2f ff ff ff ff ff ff ff ff ff ff ff ff .../............
75. 000004a0 ff ff ff 00 00 00 ff ff 1f fe 7f fb e7 ff 3f ff ..............?.
76. 000004b0 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff ff ff .......o........
77. 000004c0 ff ff ff ff ff ff ff 00 00 00 ff ff 5f fe ff fb ............\_...
78. 000004d0 f3 ff 1f ff ff ff ff ff ff ff fe 0f ff ff ff ff ................
79. 000004e0 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
80. 000004f0 5f fd ff fb e1 ff 1f ff ff ff ff ff ff ff fe a7 \_...............
81. 00000500 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
82. 00000510 00 00 ff ff df fd ff fb e1 ff 0f ff ff ff ff ff ................
83. 00000520 ff ff fe cf ff ff ff ff ff ff ff ff ff ff ff ff ................
84. 00000530 ff ff ff 00 00 00 ff fe 0f ff ff f7 e1 fe 07 ff ................
85. 00000540 ff ff ff ff ff ff fe 6f ff ff ff ff ff 80 1f ff .......o........
86. 00000550 ff ff ff ff ff ff ff 00 00 00 ff fe 9f ff ff e7 ................
87. 00000560 f5 c0 f7 ff ff ff ff ff ff ff fe ef ff ff ff ff ................
88. 00000570 e0 3d c3 ff ff ff ff ff ff ff ff 00 00 00 ff fe .=..............
89. 00000580 9f ff ff ef ff ff f7 ff ff ff ff ff ff ff fe ef ................
90. 00000590 ff ff ff ff 40 3f 38 ff ff ff ff ff ff ff ff 00 ....@?8.........
91. 000005a0 00 00 ff fe 9f ff ff df ff ff f7 ff ff ff ff ff ................
92. 000005b0 ff ff fe 6f ff ff ff ff 80 bf 9e 1f ff ff ff ff ...o............
93. 000005c0 ff ff ff 00 00 00 ff fe 1f df ff df f7 ff f7 ff ................
94. 000005d0 ff ff ff ff ff ff fe ef ff ff ff ff c2 1f ff ef ................
95. 000005e0 ff ff ff ff ff ff ff 00 00 00 ff fe cf df ff 3f ...............?
96. 000005f0 f3 ff ef ff ff ff ff ff ff ff fe ef ff ff ff ff ................
97. 00000600 c1 8f df e7 ff ff ff ff ff ff ff 00 00 00 ff fe ................
98. 00000610 af bf ff ff fb ff ef ff ff ff ff ff ff ff fe ef ................
99. 00000620 ff ff ff ff c3 ee af eb ff ff ff ff ff ff ff 00 ................
100. 00000630 00 00 ff fe af 7f f8 ff fd ff cf ff ff ff ff ff ................
101. 00000640 ff ff fc 6f ff ff ff ff e0 e2 ff e1 ff ff ff ff ...o............
102. 00000650 ff ff ff 00 00 00 ff fe ee ff f7 ff f9 ff df ff ................
103. 00000660 ff ff ff ff ff ff fe 6f ff ff ff ff f8 ff 1f fc .......o........
104. 00000670 ff ff ff ff ff ff ff 00 00 00 ff fe ec ff ef ff ................
105. 00000680 fb ff bf ff ff ff ff ff ff ff fe 6f ff ff ff ff ...........o....
106. 00000690 fc 3c cf 7e ff ff ff ff ff ff ff 00 00 00 ff fe .<.~............
107. 000006a0 c9 ff df ff f7 ff 7f ff ff ff ff ff ff ff fe ef ................
108. 000006b0 ff ff ff ff fc 20 ef be 7f ff ff ff ff ff ff 00 ..... ..........
109. 000006c0 00 00 ff fc 83 ff bf ff e7 fe 7f ff ff ff ff ff ................
110. 000006d0 ff ff fe 0f ff ff ff ff fe 03 f7 ef 3f ff ff ff ............?...
111. 000006e0 ff ff ff 00 00 00 ff fd 8f ff bf ff ef ff ff ff ................
112. 000006f0 ff ff ff ff ff ff fe af ff ff ff ff ff af f2 ff ................
113. 00000700 bf ff ff ff ff ff ff 00 00 00 ff f9 cf fc 7f ff ................
114. 00000710 ef fe ff ff ff ff ff ff ff ff fe cf ff ff ff ff ................
115. 00000720 ff bf f8 77 9f ff ff ff ff ff ff 00 00 00 ff fb ...w............
116. 00000730 bf fd ff ff 7f fe ff ff ff ff ff ff ff ff fe ef ................
117. 00000740 ff ff ff ff ff ff b8 3f ef ff ff ff ff ff ff 00 .......?........
118. 00000750 00 00 ff f9 7f fb ff fe ff ff ff ff ff ff ff ff ................
119. 00000760 ff ff fe cf ff ff ff ff ff ff 81 1d e7 ff ff ff ................
120. 00000770 ff ff ff 00 00 00 ff f8 ff f3 ff f8 ff fe ff ff ................
121. 00000780 ff ff ff ff ff ff fe ef ff ff ff ff ff ff 80 0d ................
122. 00000790 f7 ff ff ff ff ff ff 00 00 00 ff fb ff e7 ff e7 ................
123. 000007a0 ff fd ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
124. 000007b0 ff ff 80 8f fb ff ff ff ff ff ff 00 00 00 ff f7 ................
125. 000007c0 ff ef ff df ff fb ff ff ff ff ff ff ff ff fe cf ................
126. 000007d0 ff ff ff ff ff ff c3 e6 fb ff ff ff ff ff ff 00 ................
127. 000007e0 00 00 ff f7 ff ef ff 3f ff fb ff ff ff ff ff ff .......?........
128. 000007f0 ff ff fe ef ff ff ff ff ff ff e3 f7 7b ff ff ff ............{...
129. 00000800 ff ff ff 00 00 00 ff f7 ff ef f2 7f ff fb ff ff ................
130. 00000810 ff ff ff ff ff ff fe ef ff ff ff ff ff ff f0 f0 ................
131. 00000820 fc ff ff ff ff ff ff 00 00 00 ff f7 ff ef e7 ff ................
132. 00000830 ff bf ff ff ff ff ff ff ff ff ff ef ff ff ff ff ................
133. 00000840 ff ff fa f5 5e 7f ff ff ff ff ff 00 00 00 ff e7 ....^...........
134. 00000850 ff df bf ff fc ff ff ff ff ff ff ff ff ff fe ef ................
135. 00000860 ff ff ff ff ff ff fc 21 bf 3f ff ff ff ff ff 00 .......!.?......
136. 00000870 00 00 ff ef ff 5d ff ff fd ff ff ff ff ff ff ff .....]..........
137. 00000880 ff ff ff 6f ff ff ff ff ff ff fe 38 bf df ff ff ...o.......8....
138. 00000890 ff ff ff 00 00 00 ff ef ff c3 ff ff f3 ff ff ff ................
139. 000008a0 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff fe 0f .......o........
140. 000008b0 37 df ff ff ff ff ff 00 00 00 ff ef ff 87 bf ff 7...............
141. 000008c0 e7 ff ff ff ff ff ff ff ff ff ff 2f ff ff ff ff .........../....
142. 000008d0 ff ff ff ff eb ef ff ff ff ff ff 00 00 00 ff ef ................
143. 000008e0 ff 62 7f ff 1f ff ff ff ff ff ff ff ff ff fe 0f .b..............
144. 000008f0 ff ff ff ff ff ff ff ff ab e7 ff ff ff ff ff 00 ................
145. 00000900 00 00 ff df ff fc 3f fe 7f ff ff ff ff ff ff ff ......?.........
146. 00000910 ff ff fe 4f ff ff ff ff ff ff ff ff c5 f3 ff ff ...O............
147. 00000920 ff ff ff 00 00 00 ff df ff ff c0 f9 ff ff ff ff ................
148. 00000930 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff ff ff .......o........
149. 00000940 f3 fc ff ff ff ff ff 00 00 00 ff df ff ff ff 6f ...............o
150. 00000950 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
151. 00000960 ff ff ff ff ff ff 7f ff ff ff ff 00 00 00 ff ff ................
152. 00000970 ff ff ff 9f ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
153. 00000980 ff ff ff ff ff ff ff ff f7 ff 7f ff ff ff ff 00 ................
154. 00000990 00 00 ff ff ff ff ff e3 ff ff ff ff ff ff ff ff ................
155. 000009a0 ff ff ff 6f ff ff ff ff ff ff ff ff e1 ff 7f ff ...o............
156. 000009b0 ff ff ff 00 00 00 ff df ff ff ff fc 7f ff ff ff ................
157. 000009c0 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff ff ff .......o........
158. 000009d0 e1 ff 7f ff ff ff ff 00 00 00 ff ef ff ff ff ff ................
159. 000009e0 87 ff ff ff ff ff ff ff ff ff fe ef ff ff ff ff ................
160. 000009f0 ff ff ff ff cf f7 7f ff ff ff ff 00 00 00 ff f3 ................
161. 00000a00 ff ff ff ff f8 3f ff ff ff ff ff ff ff ff fe 6f .....?.........o
162. 00000a10 ff ff ff ff ff ff fe 00 3f fe 7f ff ff ff ff 00 ........?.......
163. 00000a20 00 00 ff ed ff ff ff ff ff bf ff ff ff ff ff ff ................
164. 00000a30 ff ff fe ef ff ff ff ff ff ff f3 ff ff fe ff ff ................
165. 00000a40 ff ff ff 00 00 00 ff ee 7f ff ff ff ff cf ff ff ................
166. 00000a50 ff ff ff ff ff ff fe 6f ff ff ff ff ff ff 9f ff .......o........
167. 00000a60 ff fe ff ff ff ff ff 00 00 00 ff 9b 9f ff ff ff ................
168. 00000a70 ff f3 ff ff ff ff ff ff ff ff fc 6f ff ff ff ff ...........o....
169. 00000a80 ff f2 7f ff ff de 7f ff ff ff ff 00 00 00 ff bb ................
170. 00000a90 ef ff ff ff ff fb ff ff ff ff ff ff ff ff ff 6f ...............o
171. 00000aa0 ff ff ff ff ff cf ff ff ff fd 7f ff ff ff ff 00 ................
172. 00000ab0 00 00 ff bf ff ff ff ff ff fb ff ff ff ff ff ff ................
173. 00000ac0 ff ff fe 2f ff ff ff ff ff bf ff ff ff fd bf ff .../............
174. 00000ad0 ff ff ff 00 00 00 ff a7 f1 ff ff ff ff f9 ff ff ................
175. 00000ae0 ff ff ff ff ff ff fe 0f ff ff ff ff c0 ff f7 ff ................
176. 00000af0 ff e3 8f ff ff ff ff 00 00 00 ff b7 f6 7f ff ff ................
177. 00000b00 ff ff ff ff ff ff ff ff ff ff fe 6f ff ff ff fe ...........o....
178. 00000b10 3f ff ff ff ff d3 ff ff ff ff ff 00 00 00 ff b7 ?...............
179. 00000b20 ff 7f ff ff ff fe ff ff ff ff ff ff ff ff fe 6f ...............o
180. 00000b30 ff ff ff fc ff ff ff ff ff 99 ef ff ff ff ff 00 ................
181. 00000b40 00 00 ff a7 ff 8f ff ff ff fe 7f ff ff ff ff ff ................
182. 00000b50 ff ff fe 6f ff ff ff e1 ff ff ff ff fe 0c e7 ff ...o............
183. 00000b60 ff ff ff 00 00 00 ff bf bf cf ff ff ff ff 1f ff ................
184. 00000b70 ff ff ff ff ff ff fe 6f ff ff ff cf ff ff ff ff .......o........
185. 00000b80 fd ee f3 ff ff ff ff 00 00 00 ff bf f7 e7 ff ff ................
186. 00000b90 ff ff 8f ff ff ff ff ff ff ff ff ef ff ff ff df ................
187. 00000ba0 ff ff ff ff fb f7 7b ff ff ff ff 00 00 00 ff bf ......{.........
188. 00000bb0 d7 e7 ff ff ff ff 87 ff ff ff ff ff ff ff fe ef ................
189. 00000bc0 ff ff ff 1f ff ff ff ff f3 f7 fd ff ff ff ff 00 ................
190. 00000bd0 00 00 ff 9b d3 e1 ff ff ff ff c3 ff ff ff ff ff ................
191. 00000be0 ff ff fe 6f ff ff f8 7f ff ff ff ff 83 f1 7c ff ...o..........|.
192. 00000bf0 ff ff ff 00 00 00 ff db db fd ff ff ff ff e3 ff ................
193. 00000c00 ff ff ff ff ff ff fe 6f ff ff fb ff ff ff ff ff .......o........
194. 00000c10 bc 00 7e ff ff ff ff 00 00 00 ff eb db fd ff ff ..~.............
195. 00000c20 ff ff e4 ff ff ff ff ff ff ff fe 6f ff ff e7 ff ...........o....
196. 00000c30 ff ff ff fc 7f f7 fe ff ff ff ff 00 00 00 ff ff ................
197. 00000c40 df fc ff ff ff ff e6 ff ff ff ff ff ff ff fe 67 ...............g
198. 00000c50 ff ff df ff ff ff ff fd ff f7 fe ff ff ff ff 00 ................
199. 00000c60 00 00 ff ef 9f fe 7f ff ff ff f7 7f ff ff ff ff ................
200. 00000c70 ff ff fe 27 ff ff 9f ff ff ff ff f7 ff ff ff ff ...'............
201. 00000c80 ff ff ff 00 00 00 ff eb df fd ff ff ff ff e7 bf ................
202. 00000c90 ff ff ff ff ff ff fe 6f ff ff 7f ff ff ff ff ef .......o........
203. 00000ca0 ff ff fd ff ff ff ff 00 00 00 ff eb df fd ff ff ................
204. 00000cb0 ff ff f1 bf ff ff ff ff ff ff fe 2f ff fe 7f ff .........../....
205. 00000cc0 ff ff ff 9f ff ff fd ff ff ff ff 00 00 00 ff eb ................
206. 00000cd0 cb fb ff ff ff ff e0 0f ff ff ff c0 04 0f fe 0f ................
207. 00000ce0 ff fc ff ff ff ff fe 7f ff ff ed ff ff ff ff 00 ................
208. 00000cf0 00 00 ff eb eb f7 ff ff ff ff e8 2f ff ff c0 1f .........../....
209. 00000d00 ff e0 1e 6f ff f1 ff ff ff ff fc ff ff ff fd ff ...o............
210. 00000d10 ff ff ff 00 00 00 ff e7 f3 ef ff ff ff ff f7 03 ................
211. 00000d20 ff fe 3f ff ff ff c0 2f ff ef ff ff ff ff fb ff ..?..../........
212. 00000d30 ff ff fb ff ff ff ff 00 00 00 ff ef f7 ef ff ff ................
213. 00000d40 ff ff ff 80 ff 00 ff ff ff ff f8 0e 3f df ff ff ............?...
214. 00000d50 ff ff e7 ff ff ff ef ff ff ff ff 00 00 00 ff f7 ................
215. 00000d60 f7 ff ff ff ff ff ff e0 00 0f ff ff ff ff fe 0c ................
216. 00000d70 07 df ff ff ff ff 87 ff ff ff cf ff ff ff ff 00 ................
217. 00000d80 00 00 ff d7 f7 ef ff ff ff ff ff ff ff ff ff ff ................
218. 00000d90 ff ff ff e0 00 3f ff ff ff f8 0f ff bf f0 3f ff .....?........?.
219. 00000da0 ff ff ff 00 00 00 ff d7 f7 ff ff ff ff ff ff ff ................
220. 00000db0 ff ff ff ff ff ff ff fa 36 ff ff ff ff f9 db ff ........6.......
221. 00000dc0 ff cf ff ff ff ff ff 00 00 00 ff d7 e3 df ff ff ................
222. 00000dd0 ff ff ff ff ff ff ff ff ff ff ff e0 00 ff ff ff ................
223. 00000de0 ff f8 07 ff f8 9f ff ff ff ff ff 00 00 00 ff ff ................
224. 00000df0 f3 ff ff ff ff ff ff ff ff ff ff ff ff ff ff e0 ................
225. 00000e00 09 ff ff ff ff ff ff ff 8f ff ff ff ff ff ff 00 ................
226. 00000e10 00 00 ff f3 fb df ff ff ff ff ff ff ff ff ff ff ................
227. 00000e20 ff ff ff dc 0d ff ff ff ff ff fb fc ff ff ff ff ................
228. 00000e30 ff ff ff 00 00 00 ff b9 fb df ff ff ff ff ff ff ................
229. 00000e40 ff ff ff ff ff ff ff 8f 9b ff ff ff ff ff fb 81 ................
230. 00000e50 ff ff ff ff ff ff ff 00 00 00 ff fc f5 ff ff ff ................
231. 00000e60 ff ff ff ff ff ff ff ff ff ff ff c8 db ff ff ff ................
232. 00000e70 ff ff fa 7f ff ff ff ff ff ff ff 00 00 00 ff fe ................
233. 00000e80 f7 ff ff ff ff ff ff ff ff ff ff ff ff ff ff 88 ................
234. 00000e90 3b ff ff ff ff ff f8 ff ff ff ff ff ff ff ff 00 ;...............
235. 00000ea0 00 00 ff de 6e bf ff ff ff ff ff ff ff ff ff ff ....n...........
236. 00000eb0 ff ff ff 80 33 ff ff ff ff ff f9 ff ff ff ff ff ....3...........
237. 00000ec0 ff ff ff 00 00 00 ff df af bf ff ff ff ff ff ff ................
238. 00000ed0 ff ff ff ff ff ff ff 94 77 ff ff ff ff ff fb ff ........w.......
239. 00000ee0 ff ff ff ff ff ff ff 00 00 00 ff df af ff ff ff ................
240. 00000ef0 ff ff ff ff ff ff ff ff ff ff ff 14 7f ff ff ff ................
241. 00000f00 ff ff fb ff ff ff ff ff ff ff ff 00 00 00 ff df ................
242. 00000f10 ae ff ff ff ff ff ff ff ff ff ff ff ff ff ff 28 ...............(
243. 00000f20 7f ff ff ff ff ff f9 ff ff ff ff ff ff ff ff 00 ................
244. 00000f30 00 00 ff ff ef ff ff ff ff ff ff ff ff ff ff ff ................
245. 00000f40 ff ff ff 7e ff ff ff ff ff ff fd ff ff ff ff ff ...~............
246. 00000f50 ff ff ff 00 00 00 ff ef 6d ff ff ff ff ff ff ff ........m.......
247. 00000f60 ff ff ff ff ff ff fe fd ff ff ff ff ff ff fd ff ................
248. 00000f70 ff ff ff ff ff ff ff 00 00 00 ff ef 6d ff ff ff ............m...
249. 00000f80 ff ff ff ff ff ff ff ff ff ff fe fd ff ff ff ff ................
250. 00000f90 ff ff fd ff ff ff ff ff ff ff ff 00 00 00 ff f2 ................
251. 00000fa0 ed ff ff ff ff c7 ff ff ff ff ff ff ff ff fd fd ................
252. 00000fb0 ff ff ff ff ff ff fc ff ff ff ff ff ff ff ff 00 ................
253. 00000fc0 00 00 ff fb fd ff ff ff ff c3 ff ff ff ff ff ff ................
254. 00000fd0 ff ff fd fd ff ff ff ff ff ff fc ff ff ff ff ff ................
255. 00000fe0 ff ff ff 00 00 00 ff f5 df ff ff ff ff e8 ff 9f ................
256. 00000ff0 ff ff ff ff ff ff fb fb ff ff ff ff ff ff fe ff ................
257. 00001000 ff ff ff ff ff ff ff 00 00 00 ff fe ............
258. 16 -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x16
259. class = TS\_SECONDARY (0x02)
260. orderType = TS\_ALTSEC\_GDIP\_FIRST (0x05)
261. 24 -> DRAW\_GDIPLUS\_FIRST\_ORDER::pad1Octet
262. 00 10 -> DRAW\_GDIPLUS\_FIRST\_ORDER::cbSize = 0x1000 = 4096 bytes
263. 24 22 00 00 -> DRAW\_GDIPLUS\_FIRST\_ORDER::cbTotalSize = 0x00002224 = 8740 bytes
264. 24 22 00 00 -> DRAW\_GDIPLUS\_FIRST\_ORDER::cbTotalEmfSize = 0x00002224 = 8740 bytes
265. The next 4096 bytes are the EMF+ Records (DRAW\_GDIPLUS\_FIRST\_ORDER::emfRecords). These
266. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

### Draw GDI+ Next

The following is an annotated dump of a [Draw GDI+ Next (section 2.2.2.2.1.3.6.6)](#Section_b5deca1d5802428998cb87035e1ca454) Alternate Secondary Drawing Order.

1. 00000000 1a ef 00 10 df bf ff ff ff ea 0f 8f ff ff ff ff ................
2. 00000010 ff ff fb fb ff ff ff ff ff ff fe ff ff ff ff ff ................
3. 00000020 ff ff ff 00 00 00 ff f6 df bf ff ff ff eb e3 af ................
4. 00000030 ff ff ff ff ff ff f9 f3 ff ff ff ff ff ff c2 ff ................
5. 00000040 ff ff ff ff ff ff ff 00 00 00 ff f7 5e bf ff ff ............^...
6. 00000050 ff eb f0 2f ff ff ff ff ff ff fb f7 ff ff ff ff .../............
7. 00000060 ff ff 18 ff ff ff ff ff ff ff ff 00 00 00 ff fb ................
8. 00000070 3e ff ff ff ff e9 fc 23 ff ff ff ff ff ff fb f7 >......#........
9. 00000080 ff ff ff ff ff 81 d8 ff ff ff ff ff ff ff ff 00 ................
10. 00000090 00 00 ff fd ac df ff ff ff e6 00 00 7f ff 0f ff ................
11. 000000a0 ff ff f0 67 1f ff ff ff f8 3e 02 ff ff ff ff ff ...g.....>......
12. 000000b0 ff ff ff 00 00 00 ff fe fe ff ff ff ff e3 3f 2e ..............?.
13. 000000c0 1f ff 60 00 00 00 00 00 5f ff ff ff 82 80 01 ff ..`.....\_.......
14. 000000d0 ff ff ff ff ff ff ff 00 00 00 ff ff df 7f ff ff ................
15. 000000e0 ff 80 1f 47 cf ff 7f ff ff ff ff ff 5f ff ff fe ...G........\_...
16. 000000f0 61 0f fc ff ff ff ff ff ff ff ff 00 00 00 ff fe a...............
17. 00000100 ff 6f ff ff ff 99 cb e3 e7 ff 7f fd ff ff ff fb .o..............
18. 00000110 9f ff ff f8 fd 3f fd ff ff ff ff ff ff ff ff 00 .....?..........
19. 00000120 00 00 ff ff ff 0f ff ff ff c9 e6 e9 f9 ff 3f ff ..............?.
20. 00000130 7f ff d5 7f bf ff ff e6 b8 7f fd ff ff ff ff ff ................
21. 00000140 ff ff ff 00 00 00 ff ff 2f cf ff ff ff e1 80 e0 ......../.......
22. 00000150 58 1f 38 80 00 00 00 03 bf ff fe 07 03 ff f9 ff X.8.............
23. 00000160 ff ff ff ff ff ff ff 00 00 00 ff ff bf ff ff ff ................
24. 00000170 ff f9 e8 60 14 c7 39 f8 7f ff fc 83 bf ff fc c2 ...`..9.........
25. 00000180 3f ff fb ff ff ff ff ff ff ff ff 00 00 00 ff ff ?...............
26. 00000190 cf df ff ff ff f9 c3 87 c0 01 39 00 01 ff ff f3 ..........9.....
27. 000001a0 3f ff 80 c0 8f ff fb ff ff ff ff ff ff ff ff 00 ?...............
28. 000001b0 00 00 ff ff ef df ff ff ff fc e3 ee 1b 74 10 3a .............t.:
29. 000001c0 00 3f ff f7 7f ff 2f 1c 03 ff f7 ff ff ff ff ff .?..../.........
30. 000001d0 ff ff ff 00 00 00 ff ff f7 c7 ff ff ff fe e1 c0 ................
31. 000001e0 4b 70 1c 01 00 0f ff f7 7f fc 78 30 13 ff e7 ff Kp........x0....
32. 000001f0 ff ff ff ff ff ff ff 00 00 00 ff ff fa 07 ff ff ................
33. 00000200 ff fe 0f 83 c3 78 1c 00 00 03 ff f7 7f f9 f9 84 .....x..........
34. 00000210 18 4f e7 ff ff ff ff ff ff ff ff 00 00 00 ff ff .O..............
35. 00000220 ff f3 ff ff ff ff f7 3f e0 60 00 11 00 00 ff f7 .......?.`......
36. 00000230 7f e3 f2 14 18 0f ef ff ff ff ff ff ff ff ff 00 ................
37. 00000240 00 00 ff ff ff ff ff ff ff ff f2 7f e3 e0 00 00 ................
38. 00000250 00 00 ff f7 7f cf e4 12 09 e7 ef ff ff ff ff ff ................
39. 00000260 ff ff ff 00 00 00 ff ff ff fd ff ff ff ff ff ff ................
40. 00000270 e3 c0 00 00 00 00 1f f7 7e 1c 00 60 00 27 df ff ........~..`.'..
41. 00000280 ff ff ff ff ff ff ff 00 00 00 ff ff ff fe ff ff ................
42. 00000290 ff ff ff ff f0 04 01 ea 60 01 df f7 7c f3 f7 ea ........`...|...
43. 000002a0 1d 3f ff ff ff ff ff ff ff ff ff 00 00 00 ff ff .?..............
44. 000002b0 ff ff 7f ff ff ff ff ff f0 0c 00 00 01 e7 cf f7 ................
45. 000002c0 73 e7 e3 c6 0d 7f bf ff ff ff ff ff ff ff ff 00 s...............
46. 000002d0 00 00 ff ff ff ff bf ff ff ff ff ff f0 08 0c 41 ...............A
47. 000002e0 41 83 cf f7 07 9f f0 13 98 ff bf ff ff ff ff ff A...............
48. 000002f0 ff ff ff 00 00 00 ff ff ff ff bf ff ff ff ff ff ................
49. 00000300 fe 10 7b ff f7 ff ef f7 7e 7f fc 09 3b ff bf ff ..{.....~...;...
50. 00000310 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff bf ff ................
51. 00000320 ff ff ff ff ff 9e 7f ff ff fe c7 f7 7e ff ff 00 ............~...
52. 00000330 03 ff bf ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
53. 00000340 ff ff df ff ff ff ff ff ff 97 ff ff bf ff c7 f7 ................
54. 00000350 00 ff ff 8e 5f fe 3f ff ff ff ff ff ff ff ff 00 ....\_.?.........
55. 00000360 00 00 ff ff ff ff e7 ff ff ff ff ff ff cf 7f ff ................
56. 00000370 ff fe 57 f6 07 ff ff e0 00 00 3f ff ff ff ff ff ..W.......?.....
57. 00000380 ff ff ff 00 00 00 ff ff ff ff fb ff ff ff ff ff ................
58. 00000390 fe ce ff ff ff ff f7 f6 ff ff ff 80 ff ff 3f ff ..............?.
59. 000003a0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff 7f ................
60. 000003b0 ff ff ff ff ff ce ff ff ff ff f7 f6 ff ff c0 02 ................
61. 000003c0 ff ff 3f ff ff ff ff ff ff ff ff 00 00 00 ff ff ..?.............
62. 000003d0 ff ff ff 7f ff ff ff ff ff cf ff ff ff ff f7 f6 ................
63. 000003e0 ff fc 3f fe d0 3c 3f ff ff ff ff ff ff ff ff 00 ..?..<?.........
64. 000003f0 00 00 ff ff ff ff ff bf ff ff ff ff ff cf ff ff ................
65. 00000400 ff ff f7 f6 ff e1 ff 54 05 ff 3f ff ff ff ff ff .......T..?.....
66. 00000410 ff ff ff 00 00 00 ff ff ff ff ff df ff ff ff ff ................
67. 00000420 ff df ff ff ff ff e7 f6 7e 0f ff c0 ff ff 7f ff ........~.......
68. 00000430 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff e3 ................
69. 00000440 ff ff ff ff ff 0f ff ff ff ff f7 f4 80 ff c0 0f ................
70. 00000450 ff e7 7f ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
71. 00000460 ff ff ff fc ff ff ff ff f8 3f ff ff ff ff c7 e4 .........?......
72. 00000470 7f c0 7c 7f ff f5 7f ff ff c0 3f ff ff ff ff 00 ..|.......?.....
73. 00000480 00 00 ff ff ff ff ff fe 3f ff ff ff c3 7f ff ff ........?.......
74. 00000490 ff ff e7 ec 60 0f fd 3f ff fd 7f ff fe 03 cf ff ....`..?........
75. 000004a0 ff ff ff 00 00 00 ff ff ff ff ff ff cf ff ff fc ................
76. 000004b0 1e ff ff ff ff ff c7 e4 03 ff f9 bf ff fe 7f ff ................
77. 000004c0 f8 7f ef ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
78. 000004d0 e3 ff ff e0 c9 ff ff ff ff ff e7 ec 3f ff f9 bf ............?...
79. 000004e0 fd ff 7f ff fb df ef ff ff ff ff 00 00 00 ff ff ................
80. 000004f0 ff ff ff ff fd ff ff 81 c7 ff fd fb 7f b0 a7 e4 ................
81. 00000500 7f ff f1 bf ff ff 7f ff f3 97 ff ff ff ff ff 00 ................
82. 00000510 00 00 ff ff ff ff ff ff ff 00 40 0c 1f e9 bf 5a ..........@....Z
83. 00000520 06 80 03 ee ff ff 0f 7f ff ff 7f 0f c3 87 ef ff ................
84. 00000530 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff 00 00 ................
85. 00000540 fb 00 40 00 40 0c 03 ee ff 00 fc ff fb ff 7f 21 ..@.@..........!
86. 00000550 d7 c7 ef ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
87. 00000560 ff ff 80 ff ff 20 00 02 00 04 03 e6 70 3f f9 ff ..... ......p?..
88. 00000570 ff ff 7f 48 d7 ff ef ff ff ff ff 00 00 00 ff ff ...H............
89. 00000580 ff ff ff ff ff ff 8f e1 d0 00 00 00 40 24 0b e3 ............@$..
90. 00000590 77 ff 73 ff ff ff 3f 78 e7 ff f7 ff ff ff ff 00 w.s...?x........
91. 000005a0 00 00 ff ff ff ff ff ff ff ff 15 8b 40 00 00 00 ............@...
92. 000005b0 00 00 9b e9 47 fe 73 ff fb ff 3e 6d 4f ff fb ff ....G.s...>mO...
93. 000005c0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff 00 04 ................
94. 000005d0 00 00 00 00 00 00 13 b4 0f fc c7 ff fc ff 3e 2c ..............>,
95. 000005e0 1f ff f9 ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
96. 000005f0 ff ff 20 00 00 7f ff ff f8 00 13 1b bf f1 8f ff .. .............
97. 00000600 e7 ff be 70 1f ff ed ff ff ff ff 00 00 00 ff ff ...p............
98. 00000610 ff ff ff ff ff ff a0 20 1f ff ff ff fe 00 10 00 ....... ........
99. 00000620 3f c7 3f ff c7 ff be 3c 9f ff ef ff ff ff ff 00 ?.?....<........
100. 00000630 00 00 ff ff ff ff ff ff ff ff 80 00 ff ff ff ff ................
101. 00000640 ff 00 06 0e 3e 1e 3f ff c1 ff be 1c 07 ff fc ff ....>.?.........
102. 00000650 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff e0 47 ...............G
103. 00000660 ff ff ff ff ff e3 ee 6f 7e 7c ff ff af ff bf 8d .......o~|......
104. 00000670 d7 fe fe ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
105. 00000680 ff ff e8 0f ff ff ff ff ff f0 5e 6f 7e fc ff ff ..........^o~...
106. 00000690 af ff bf cd d7 fe fe ff ff ff ff 00 00 00 ff ff ................
107. 000006a0 ff ff ff ff ff ff f3 8f ff ff ff ff ff fc 3e 6f ..............>o
108. 000006b0 78 fc ff fe 27 ff bf e4 d7 fc ff 7f ff ff ff 00 x...'...........
109. 000006c0 00 00 ff ff ff ff ff ff ff ff fc 3f ff ff ff ff ...........?....
110. 000006d0 ff ff fe 6f 3b fc ff fe f7 ff bf f5 f7 fd ff 7f ...o;...........
111. 000006e0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
112. 000006f0 ff ff ff ff ff ff fe 6f 33 fc 3f f8 e7 ff bf f5 .......o3.?.....
113. 00000700 f7 fd fd 7f ff ff ff 00 00 00 ff ff ff ff ff ff ................
114. 00000710 ff ff ff ff ff ff ff ff ff ff fe 6f 87 fc 03 e3 ...........o....
115. 00000720 f7 ff 9f f4 37 fb ff ff ff ff ff 00 00 00 ff ff ....7...........
116. 00000730 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 6f ...............o
117. 00000740 87 81 f0 07 fb ff df f8 a7 ff fd bf ff ff ff 00 ................
118. 00000750 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
119. 00000760 ff ff fe 6f f7 3f ff ff fb ff cf f8 b7 ff ff bf ...o.?..........
120. 00000770 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
121. 00000780 ff ff ff ff ff ff fe 2f f6 7f ff ff f9 ff ef f9 ......./........
122. 00000790 37 ff ff bf ff ff ff 00 00 00 ff ff ff ff ff ff 7...............
123. 000007a0 ff ff ff ff ff ff ff ff ff ff fe 2f f4 ff ff ff .........../....
124. 000007b0 fd ff e7 fd 27 ff ff bf ff ff ff 00 00 00 ff ff ....'...........
125. 000007c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 2f .............../
126. 000007d0 f0 ff ff ff fd ff f7 fd 37 ff ff bf ff ff ff 00 ........7.......
127. 000007e0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
128. 000007f0 ff ff fe 0f f1 ff fd 7f fd ff f3 fd bf ff ff df ................
129. 00000800 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
130. 00000810 ff ff ff ff ff ff fe 0f f3 ff f9 ff fd ff f3 fd ................
131. 00000820 b3 fb ff df ff ff ff 00 00 00 ff ff ff ff ff ff ................
132. 00000830 ff ff ff ff ff ff ff ff ff ff fe 6f f7 ff f9 ff ...........o....
133. 00000840 f9 ff fb fc 93 ff ff df ff ff ff 00 00 00 ff ff ................
134. 00000850 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 4f ...............O
135. 00000860 f7 ff f9 ff fb ff fb fd 13 ff fb a7 ff ff ff 00 ................
136. 00000870 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
137. 00000880 ff ff fe 6f f7 ff f9 ff fb ff f9 fd a7 fb fb 91 ...o............
138. 00000890 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
139. 000008a0 ff ff ff ff ff ff fe 6f f3 ff f9 ff f7 ff f9 fd .......o........
140. 000008b0 ad fd fd 81 ff ff ff 00 00 00 ff ff ff ff ff ff ................
141. 000008c0 ff ff ff ff ff ff ff ff ff ff ff 6f fb ff f9 ff ...........o....
142. 000008d0 e7 ff fc fd ab fd ff a3 ff ff ff 00 00 00 ff ff ................
143. 000008e0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 4f ...............O
144. 000008f0 f9 ff f9 ff cf ff fe f8 01 ff f9 d0 ff ff ff 00 ................
145. 00000900 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
146. 00000910 ff ff fe 6f fc ff c1 ff df ff fe e1 9e ff 71 14 ...o..........q.
147. 00000920 7f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
148. 00000930 ff ff ff ff ff ff ff 2f fe fc 07 fb 9f c3 ff 07 ......./........
149. 00000940 c4 ff e1 50 3f ff ff 00 00 00 ff ff ff ff ff ff ...P?...........
150. 00000950 ff ff ff ff ff ff ff ff ff ff fe 0f fe dd ff ff ................
151. 00000960 1f 9b fb 1f f5 7f e0 58 bf ff ff 00 00 00 ff ff .......X........
152. 00000970 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 2f .............../
153. 00000980 fe fd ff ff 00 33 fd 0f f1 bf e0 9c 3f ff ff 00 .....3......?...
154. 00000990 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
155. 000009a0 ff ff fe 4f fe 33 ff f0 00 e7 ff 0f f0 9f e0 0c ...O.3..........
156. 000009b0 9f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
157. 000009c0 ff ff ff ff ff ff ff 4f ff 87 ff e1 ff ef ff 0f .......O........
158. 000009d0 f9 5f f8 46 9f ff ff 00 00 00 ff ff ff ff ff ff .\_.F............
159. 000009e0 ff ff ff ff ff ff ff ff ff ff fe 4f ff cf ff c7 ...........O....
160. 000009f0 ff ef ff 07 f9 5f f8 e6 cf ff ff 00 00 00 ff ff .....\_..........
161. 00000a00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 4f ...............O
162. 00000a10 ff cf fe 0f ff cf 86 09 f8 0f f9 f0 c7 ff ff 00 ................
163. 00000a20 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
164. 00000a30 ff ff ff 6f ff df fc f0 df cf b6 09 fd 17 f9 fc ...o............
165. 00000a40 b7 ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
166. 00000a50 ff ff ff ff ff ff ff 6f ff df fd e0 3f af b7 0e .......o....?...
167. 00000a60 fc 97 ff fd bb ff ff 00 00 00 ff ff ff ff ff ff ................
168. 00000a70 ff ff ff ff ff ff ff ff ff ff fe 6f ff df f9 e6 ...........o....
169. 00000a80 03 ef 27 0f 3e 03 ff f9 bb ff ff 00 00 00 ff ff ..'.>...........
170. 00000a90 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 2f .............../
171. 00000aa0 ff df fb ef cf ef 07 47 9f 83 ff f3 bb ff ff 00 .......G........
172. 00000ab0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
173. 00000ac0 ff ff fe 2f ff df fb ef ff e7 4f 87 ef 81 ff f3 .../......O.....
174. 00000ad0 99 ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
175. 00000ae0 ff ff ff ff ff ff fe 2f ff cf fb ee ff e2 47 a7 ......./......G.
176. 00000af0 f3 c0 3f f7 99 ff ff 00 00 00 ff ff ff ff ff ff ..?.............
177. 00000b00 ff ff ff ff ff ff ff ff ff ff fe 0f ff cf f3 cf ................
178. 00000b10 1f e0 00 c1 f8 80 00 03 c8 ff ff 00 00 00 ff ff ................
179. 00000b20 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
180. 00000b30 ff cf f7 ef 9f f8 18 70 ff d6 21 79 e8 ff ff 00 .......p..!y....
181. 00000b40 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
182. 00000b50 ff ff ff 6f ff ef f7 8f 9f c0 7e fd 1f d7 07 fc ...o......~.....
183. 00000b60 c8 ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
184. 00000b70 ff ff ff ff ff ff fe 6f ff ef f7 df 3f 9e 7e fe .......o....?.~.
185. 00000b80 71 97 0f f8 4b ff ff 00 00 00 ff ff ff ff ff ff q...K...........
186. 00000b90 ff ff ff ff ff ff ff ff ff ff ff 6f ff f7 f7 fe ...........o....
187. 00000ba0 3f be fe ff 3c 57 28 1e 3b ff ff 00 00 00 ff ff ?...<W(.;.......
188. 00000bb0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
189. 00000bc0 ff f7 f7 fe ff 3e fe ff c7 c0 61 c0 13 ff ff 00 .....>....a.....
190. 00000bd0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
191. 00000be0 ff ff fe 6f ff f9 f7 f0 fc 7e fc fe 10 03 1f ff ...o.....~......
192. 00000bf0 07 ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
193. 00000c00 ff ff ff ff ff ff fe 6f ff f9 f7 e7 f9 ee fe e4 .......o........
194. 00000c10 0f f0 0f ff 5f ff ff 00 00 00 ff ff ff ff ff ff ....\_...........
195. 00000c20 ff ff ff ff ff ff ff ff ff ff ff 2f ff fd f7 c7 .........../....
196. 00000c30 fb ee fe 01 e7 f0 8f fe df ff ff 00 00 00 ff ff ................
197. 00000c40 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 2f .............../
198. 00000c50 ff fc f7 cf fb ee 7c 1d f7 fa cf fe 0f ff ff 00 ......|.........
199. 00000c60 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
200. 00000c70 ff ff ff 2f ff fe f3 cf f3 ce fc 7f f7 f9 ef df .../............
201. 00000c80 2f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff /...............
202. 00000c90 ff ff ff ff ff ff fe 2f ff ff 7b cf f7 8e 78 ff ......./..{...x.
203. 00000ca0 b7 fc 63 c3 af ff ff 00 00 00 ff ff ff ff ff ff ..c.............
204. 00000cb0 ff ff ff ff ff ff ff ff ff ff fe 4f ff ff bb cf ...........O....
205. 00000cc0 f7 9f 79 ff f7 fe e1 c0 af ff ff 00 00 00 ff ff ..y.............
206. 00000cd0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
207. 00000ce0 ff ff c1 cf e7 9f 7b f7 f7 ff 00 c0 97 ff ff 00 ......{.........
208. 00000cf0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
209. 00000d00 ff ff fe 6f ff ff f9 e7 e7 9f 73 fb f3 ff 08 32 ...o......s....2
210. 00000d10 57 ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff W...............
211. 00000d20 ff ff ff ff ff ff ff 6f ff ff fd e7 f7 9f 77 fb .......o......w.
212. 00000d30 fb ff cc 34 5f ff ff 00 00 00 ff ff ff ff ff ff ...4\_...........
213. 00000d40 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff fc e7 ...........o....
214. 00000d50 f7 9f 77 fb fb ff ec 17 1b ff ff 00 00 00 ff ff ..w.............
215. 00000d60 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
216. 00000d70 ff ff fe 20 e7 9e e7 fb f9 ff e0 0b 1b ff ff 00 ... ............
217. 00000d80 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
218. 00000d90 ff ff ff 6f ff ff ff 98 c7 9f e7 fb fc ff f8 27 ...o...........'
219. 00000da0 9f ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
220. 00000db0 ff ff ff ff ff ff ff 6f ff ff ff dc 13 9f f7 fb .......o........
221. 00000dc0 fe 7f f8 77 9b ff ff 00 00 00 ff ff ff ff ff ff ...w............
222. 00000dd0 ff ff ff ff ff ff ff ff ff ff ff 2f ff ff ff e0 .........../....
223. 00000de0 03 8f f7 f9 ff 80 00 72 03 ff ff 00 00 00 ff ff .......r........
224. 00000df0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 2f .............../
225. 00000e00 ff ff ff fe 0b ef f3 fd ff f0 1c fd 1f ff ff 00 ................
226. 00000e10 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
227. 00000e20 ff ff ff 0f ff ff ff ff fa e7 e3 fd ff ff fd fe ................
228. 00000e30 df ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
229. 00000e40 ff ff ff ff ff ff ff 0f ff ff ff ff f9 e3 fb fc ................
230. 00000e50 ff ff d9 ff 1f ff ff 00 00 00 ff ff ff ff ff ff ................
231. 00000e60 ff ff ff ff ff ff ff ff ff ff ff 4f ff ff ff ff ...........O....
232. 00000e70 fc f9 fb fe 7f ff f3 ff ff ff ff 00 00 00 ff ff ................
233. 00000e80 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
234. 00000e90 ff ff ff ff fe fc f9 ff 3f ff e7 ff ff ff ff 00 ........?.......
235. 00000ea0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
236. 00000eb0 ff ff ff 6f ff ff ff ff fe 7e f9 ff 1f ff ef ff ...o.....~......
237. 00000ec0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
238. 00000ed0 ff ff ff ff ff ff ff 6f ff ff ff ff ff 3e 3c ff .......o.....><.
239. 00000ee0 80 00 0f ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
240. 00000ef0 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
241. 00000f00 ff bf be ff ff f3 ff ff ff ff ff 00 00 00 ff ff ................
242. 00000f10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
243. 00000f20 ff ff ff ff ff 8f 82 3f ff cf ff ff ff ff ff 00 .......?........
244. 00000f30 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
245. 00000f40 ff ff ff 6f ff ff ff ff ff e7 fb 8f ff df ff ff ...o............
246. 00000f50 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
247. 00000f60 ff ff ff ff ff ff ff 6f ff ff ff ff ff f0 f8 e7 .......o........
248. 00000f70 ff 9f ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
249. 00000f80 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
250. 00000f90 ff fc 7e e1 ff 9f ff ff ff ff ff 00 00 00 ff ff ..~.............
251. 00000fa0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 2f .............../
252. 00000fb0 ff ff ff ff ff ff 02 7b fc 3f ff ff ff ff ff 00 .......{.?......
253. 00000fc0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
254. 00000fd0 ff ff ff 0f ff ff ff ff ff ff fc 1f f8 ff ff ff ................
255. 00000fe0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
256. 00000ff0 ff ff ff ff ff ff ff 0f ff ff ff ff ff ff ff c0 ................
257. 00001000 03 ff ff ff ....
258. 1a -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x1a
259. class = TS\_SECONDARY (0x02)
260. orderType = TS\_ALTSEC\_GDIP\_NEXT (0x06)
261. ef -> DRAW\_GDIPLUS\_NEXT\_ORDER::pad1Octet
262. 00 10 -> DRAW\_GDIPLUS\_NEXT\_ORDER::cbSize = 0x1000 = 4096 bytes
263. The next 4096 bytes are the EMF+ Records (DRAW\_GDIPLUS\_NEXT\_ORDER::emfRecords). These
264. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

### Draw GDI+ End

The following is an annotated dump of a [Draw GDI+ End (section 2.2.2.2.1.3.6.7)](#Section_657e77cd01d94eccaf45a69e0b48390f) Alternate Secondary Drawing Order.

1. 00000000 1e e7 24 02 24 22 00 00 24 22 00 00 ff ff ff 00 ..$.$"..$"......
2. 00000010 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
3. 00000020 ff ff ff 4f ff ff ff ff ff ff ff fc 3f ff ff ff ...O........?...
4. 00000030 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
5. 00000040 ff ff ff ff ff ff ff 6f ff ff ff ff ff ff ff ff .......o........
6. 00000050 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
7. 00000060 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
8. 00000070 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
9. 00000080 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 6f ...............o
10. 00000090 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
11. 000000a0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
12. 000000b0 ff ff fe 6f ff ff ff ff ff ff ff ff ff ff ff ff ...o............
13. 000000c0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
14. 000000d0 ff ff ff ff ff ff fe 2f ff ff ff ff ff ff ff ff ......./........
15. 000000e0 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
16. 000000f0 ff ff ff ff ff ff ff ff ff ff ff 6f ff ff ff ff ...........o....
17. 00000100 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
18. 00000110 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 2f .............../
19. 00000120 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
20. 00000130 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
21. 00000140 ff ff fe 2f ff ff ff ff ff ff ff ff ff ff ff ff .../............
22. 00000150 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
23. 00000160 ff ff ff ff ff ff fe 2f ff ff ff ff ff ff ff ff ......./........
24. 00000170 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
25. 00000180 ff ff ff ff ff ff ff ff ff ff fe 0f ff ff ff ff ................
26. 00000190 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 ff ff ................
27. 000001a0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff fe 0f ................
28. 000001b0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 ................
29. 000001c0 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
30. 000001d0 ff ff fe 4f ff ff ff ff ff ff ff ff ff ff ff ff ...O............
31. 000001e0 ff ff ff 00 00 00 ff ff ff ff ff ff ff ff ff ff ................
32. 000001f0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
33. 00000200 ff ff ff ff ff ff ff 00 00 00 ff ff ff ff ff ff ................
34. 00000210 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ................
35. 00000220 ff ff ff ff ff ff ff ff ff ff ff 00 00 00 00 00 ................
36. 1e -> ALTSEC\_DRAWING\_ORDER\_HEADER::controlFlags = 0x1e
37. class = TS\_SECONDARY (0x02)
38. orderType = TS\_ALTSEC\_GDIP\_END (0x07)
39. e7 -> DRAW\_GDIPLUS\_END\_ORDER::pad1Octet
40. 24 02 -> DRAW\_GDIPLUS\_END\_ORDER::cbSize = 0x0224 = 548 bytes
41. 24 22 00 00 -> DRAW\_GDIPLUS\_END\_ORDER::cbTotalSize = 0x00002224 = 8740 bytes
42. 24 22 00 00 -> DRAW\_GDIPLUS\_END\_ORDER::cbTotalEmfSize = 0x00002224 = 8740 bytes
43. The next 548 bytes are the EMF+ Records (DRAW\_GDIPLUS\_END\_ORDER::emfRecords). These
44. records can be parsed as specified in [MS-EMFPLUS] section 2.3.

## NineGrid Examples

The following diagram illustrates how a NineGrid bitmap may be resized when it is rendered. The [NineGrid Bitmap Information (section 2.2.2.2.1.3.4.1)](#Section_ba44b18260ad4bb199d55d00c0695bcb) structure defines the grid layout, specifically the sizes of the LeftWidth, RightWidth, TopHeight, and BottomHeight constants.

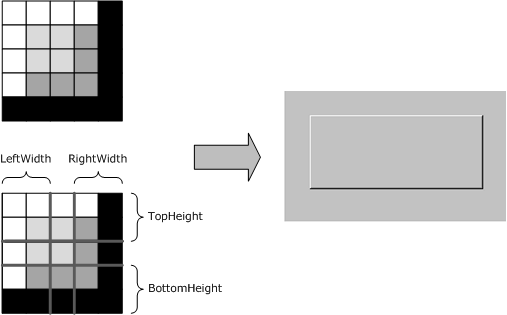


Figure : Expansion of a NineGrid bitmap

The following diagram illustrates how a source bitmap in the [NineGrid Bitmap Cache (section 3.1.1.1.6)](#Section_b71d528445c146e7b92511edd3b1de82) is clipped and then modified using a NineGrid transformation to produce a final image in a destination rectangle.

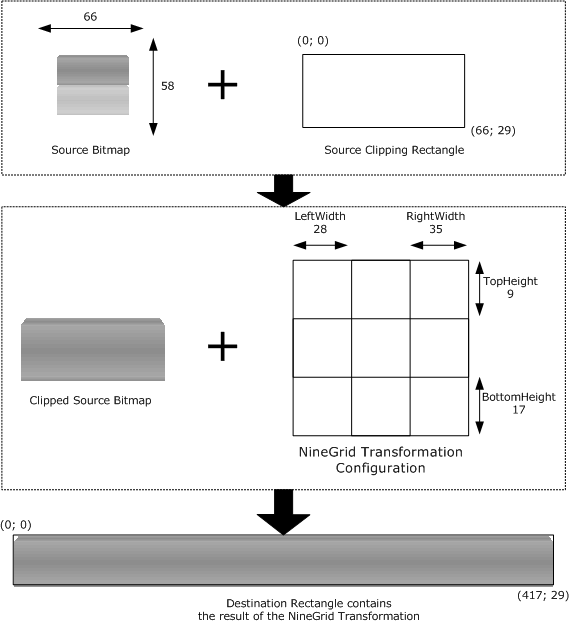


Figure : Illustration of Draw Nine Grid Primary Drawing Order

The [DrawNineGrid (section 2.2.2.2.1.1.2.21)](#Section_44bd6a68a5b74a38980706ba4ab4a72a) and [MultiDrawNineGrid (section 2.2.2.2.1.1.2.22)](#Section_101fecdc3ad841c6b7b670d8efd4ad70) Primary Drawing Orders are used to render NineGrid bitmaps that have been stored in the NineGrid Bitmap Cache using the [Create NineGrid Bitmap (section 2.2.2.2.1.3.4)](#Section_6ac5eb820c30405490e883797f24bc38) and [Stream Bitmap Orders (section 2.2.2.2.1.3.5)](#Section_5d03486bb73b463cb77330691af9dffc).

## Save Bitmap Example

The [SaveBitmap (section 2.2.2.2.1.1.2.12)](#Section_db4ce8dbcf76483d99ba354cd010ea4a) Primary Drawing Order is used to encode a rectangular screen image for saving or restoring by the client.

The following diagram shows how a 127-by-134-pixel virtual desktop rectangle received in the SaveBitmap Primary Drawing Order is tiled into the client-side [Save Bitmap (section 3.2.1.2)](#Section_c464f205346c475fb3d228fcef86a5ad), and how the position of the next bitmap can be computed.

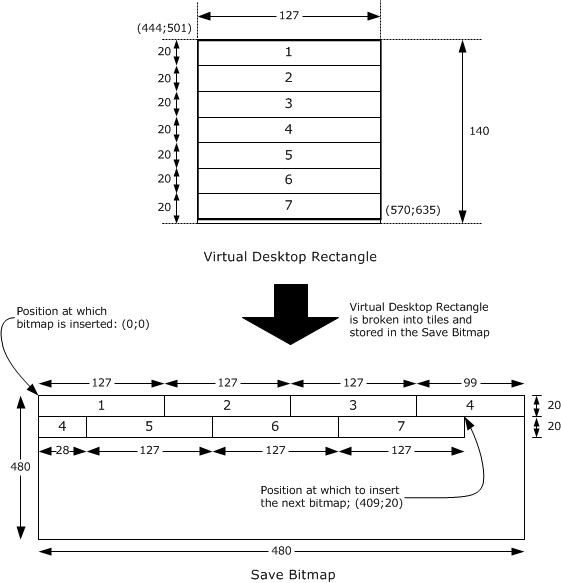


Figure : Illustration of Save Bitmap Primary Drawing Order

Applying the formulas in section 2.2.2.2.1.1.2.12 (assuming an X granularity of 1 and a Y granularity of 20):

1. AreaWidthInPixels
2. = [(width + XGranularity - 1) / XGranularity] \* XGranularity
3. = [(127 + 1 - 1) / 1] \* 1
4. = 127
5. AreaHeightInPixels
6. = [(height + YGranularity - 1) / YGranularity] \* YGranularity
7. = [(134 + 20 - 1) / 20] \* 20
8. = 140
9. Area
10. = AreaWidthInPixels \* AreaHeightInPixels
11. = 127 \* 140
12. = 17780

Hence, the area occupied by the 127-by-134-pixel bitmap in the SaveBitmap is 17,780 pixels. The X and Y position in the Save Bitmap from which to tile the next bitmap is computed using the formulas in section 2.2.2.2.1.1.2.12.

1. Y
2. = [SaveBitmapPosition / (480 \* YGranularity)] \* YGranularity
3. = [17780 / (480 \* 20)] \* 20
4. = 20
5. X
6. = [SaveBitmapPosition - (Y \* 480)] / YGranularity
7. = [17780 - (20 \* 480)] / 20
8. = 409

## Glyph Image Data

### "d" Character

The following shows glyph image data (1 bpp format) for character "d" extracted from a Cache Glyph (Revision 2) (section [2.2.2.2.1.2.6)](#Section_ffc80435136e442e908540ca1d27a413) Secondary Drawing Order.

1. Glyph width = 5 pixels
2. Glyph height = 9 pixels
3. Glyph origin = (0, -9), marked with an "X" on the image grid
4. Bitmap = { 0x08, 0x08, 0x08, 0x78, 0x88, 0x88, 0x88, 0x88, 0x78 }

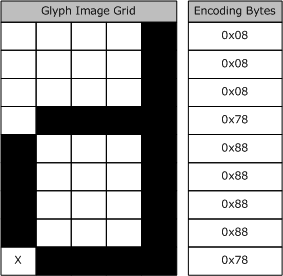


Figure : Decoded glyph bytes for "d" character

### "p" Character

The following shows glyph image data (1 bpp format) for character "p" extracted from a Cache Glyph (Revision 2) (section [2.2.2.2.1.2.6)](#Section_ffc80435136e442e908540ca1d27a413) Secondary Drawing Order.

1. Glyph width = 5 pixels
2. Glyph height = 8 pixels
3. Glyph origin = (0, -6), marked with an "X" on the image grid
4. Bitmap = { 0xF0, 0x88, 0x88, 0x88, 0x88, 0xF0, 0x80, 0x80 }

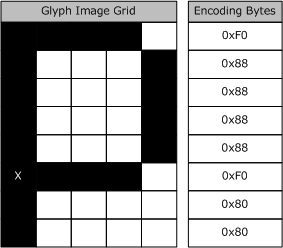


Figure : Decoded glyph bytes for "p" character

## RDP 6.1 Bulk Compression Example

The following annotation shows how the data used in the examples in section [3.1.8.2.2.1](#Section_8aa890d901dd4f7e859dd751056ebe93) and [3.1.8.2.3.1](#Section_327F048BFE254C12B9793FE20B0D15B8) appears in binary encoded format after having been decompressed by the level-2 decompressor, but not yet by the level-1 decompressor.

The first packet contains the following data.

1. a b c d e f g h i j

The encoded version of this packet is represented as follows.

1. 00000000 16 80 61 62 63 64 65 66 67 68 69 6a ..abcdefghij
2. 16 -> RDP61\_COMPRESSED\_DATA::Level1ComprFlags
3. 0x16
4. = 0x10 | 0x02 | 0x04
5. = L1\_INNER\_COMPRESSION | L1\_NO\_COMPRESSION | L1\_PACKET\_AT\_FRONT
6. 80 -> RDP61\_COMPRESSED\_DATA::Level2ComprFlags
7. 0x80
8. = PACKET\_FLUSHED
9. 61 62 63 64 65 66 67 68 69 6a -> RDP61\_COMPRESSED\_DATA::Literals

The second packet contains the following data.

1. [2 matches]
2. [MatchHistoryOffset@3, MatchLength=9, MatchOutputOffset@5]
3. [MatchHistoryOffset@0, MatchLength=4, MatchOutputOffset@14]
4. k l m n o u

The encoded version of this packet is represented as follows.

1. 00000000 11 21 02 00 09 00 05 00 03 00 00 00 04 00 0e 00 . ..............
2. 00000010 00 00 00 00 6b 6c 6d 6e 6f 75 ....klmnou
3. 11 -> RDP61\_COMPRESSED\_DATA::Level1ComprFlags
4. 0x11
5. = 0x10 | 0x01
6. = L1\_INNER\_COMPRESSION | L1\_COMPRESSED
7. 21 -> RDP61\_COMPRESSED\_DATA::Level2ComprFlags
8. 0x21
9. = PACKET\_COMPRESSED | PACKET\_COMPR\_TYPE\_64K
10. 02 00 -> RDP61\_COMPRESSED\_DATA::MatchCount = 2 matches
11. 09 00 -> RDP61\_MATCH\_DETAILS::MatchLength = 9 bytes
12. 05 00 -> RDP61\_MATCH\_DETAILS::MatchOutputOffset = 5
13. 03 00 00 00 -> RDP61\_MATCH\_DETAILS::MatchHistoryOffset = 3
14. 04 00 -> RDP61\_MATCH\_DETAILS::MatchLength = 4 bytes
15. 0e 00 -> RDP61\_MATCH\_DETAILS::MatchOutputOffset = 14
16. 00 00 00 00 -> RDP61\_MATCH\_DETAILS::MatchHistoryOffset = 0
17. 6b 6c 6d 6e 6f 75 -> RDP61\_COMPRESSED\_DATA::Literals

## Error Conditions

### Bitmap Cache Error PDU

The following is an annotated dump of a [Bitmap Cache Error PDU (section 2.2.2.3.1)](#Section_FDC28A58010549CAA9040FE3BEE58EC2).

1. 00000000 03 00 00 38 02 f0 80 64 00 06 03 eb 70 2a 28 00 ...8...d....p\*(.
2. 00000010 81 f8 05 d1 ef 10 ff fc 2e e5 0d e2 b4 57 0e 5a .............W.Z
3. 00000020 06 48 48 79 f5 08 c0 1b 2f a3 bf fc a4 16 fc f4 .HHy..../.......
4. 00000030 df 03 26 63 60 61 d0 f6 ..&c`a..
5. 03 00 00 38 -> TPKT Header (length = 56 bytes)
6. 02 f0 80 -> X.224 Data TPDU
7. 64 00 06 03 eb 70 2a -> PER encoded (ALIGNED variant of BASIC-PER) SendDataRequest
8. initiator = 1007 (0x03ef)
9. channelId = 1003 (0x03eb)
10. dataPriority = high
11. segmentation = begin | end
12. userData length = 0x2a = 42 bytes
13. 28 00 -> TS\_SECURITY\_HEADER::flags = 0x0028
14. 0x0028
15. = 0x0020 | 0x0008
16. = SEC\_IGNORE\_SEQNO | SEC\_ENCRYPT
17. 81 f8 -> TS\_SECURITY\_HEADER::flagsHi - ignored as flags field does
18. not contain SEC\_FLAGSHI\_VALID (0x8000)
19. 05 d1 ef 10 ff fc 2e e5 -> TS\_SECURITY\_HEADER1::dataSignature
20. 0d e2 b4 57 0e 5a 06 48 48 79 f5 08 c0 1b 2f a3
21. bf fc a4 16 fc f4 df 03 26 63 60 61 d0 f6 -> Encrypted TS\_BITMAP\_CACHE\_ERROR\_PDU
22. Decrypted TS\_BITMAP\_CACHE\_ERROR\_PDU:
23. 00000000 1e 00 17 00 ef 03 ea 03 01 00 00 01 c0 c0 2c 00 ..............,.
24. 00000010 00 00 01 00 00 00 02 01 00 00 00 00 00 00 ..............
25. 1e 00 -> TS\_SHARECONTROLHEADER::totalLength = 0x001e = 30 bytes
26. 17 00 -> TS\_SHARECONTROLHEADER::pduType = 0x0017
27. 0x0017
28. = 0x0010 | 0x0007
29. = TS\_PROTOCOL\_VERSION | PDUTYPE\_DATAPDU
30. ef 03 -> TS\_SHARECONTROLHEADER::pduSource = 0x03ef = 1007
31. ea 03 01 00 -> TS\_SHAREDATAHEADER::shareID = 0x000103ea
32. 00 -> TS\_SHAREDATAHEADER::pad1
33. 01 -> TS\_SHAREDATAHEADER::streamId = STREAM\_LOW (1)
34. c0 c0 -> TS\_SHAREDATAHEADER::uncompressedLength = 0xc0c0 (uninitialized due to bug)
35. 2c -> TS\_SHAREDATAHEADER::pduType2 = TS\_PDUTYPE2\_BITMAPCACHE\_ERROR\_PDU (44)
36. 00 -> TS\_SHAREDATAHEADER::generalCompressedType = 0
37. 00 00 -> TS\_SHAREDATAHEADER::generalCompressedLength = 0
38. 01 -> TS\_BITMAP\_CACHE\_ERROR\_PDU::NumInfoBlocks = 1
39. 00 -> TS\_BITMAP\_CACHE\_ERROR\_PDU::Pad1
40. 00 00 -> TS\_BITMAP\_CACHE\_ERROR\_PDU::Pad2
41. 02 -> TS\_BITMAPCACHE\_ERROR\_INFO::CacheID = 2
42. 01 -> TS\_BITMAPCACHE\_ERROR\_INFO::bBitField = 0x01 = BC\_ERR\_FLUSH\_CACHE
43. Since BC\_ERR\_NEWNUMENTRIES\_VALID (0x02) is not set, NewNumEntries is not valid.
44. 00 00 -> TS\_BITMAPCACHE\_ERROR\_INFO::Pad
45. 00 00 00 00 -> TS\_BITMAPCACHE\_ERROR\_INFO::NewNumEntries = 0

### Offscreen Bitmap Cache Error PDU

The following is an annotated dump of an [Offscreen Bitmap Cache Error PDU (section 2.2.2.3.2)](#Section_C78F94F32A7F41A8902A1C6B30D34240).

1. 00000000 03 00 00 30 02 f0 80 64 00 06 03 eb 70 22 28 00 ...0...d....p"(.
2. 00000010 81 f8 16 ff 53 cd 83 24 e0 da ad 70 d2 f8 00 9d ....S..$...p....
3. 00000020 97 29 71 0a 0e 7c cc 88 dd ac b7 53 e0 0f 3e 83 .)q..|.....S..>.
5. 03 00 00 30 -> TPKT Header (length = 48 bytes)
6. 02 f0 80 -> X.224 Data TPDU
7. 64 00 06 03 eb 70 22 -> PER encoded (ALIGNED variant of BASIC-PER) SendDataRequest
8. initiator = 1007 (0x03ef)
9. channelId = 1003 (0x03eb)
10. dataPriority = high
11. segmentation = begin | end
12. userData length = 0x22 = 34 bytes
13. 28 00 -> TS\_SECURITY\_HEADER::flags = 0x0028
14. 0x0028
15. = 0x0020 | 0x0008
16. = SEC\_IGNORE\_SEQNO | SEC\_ENCRYPT
17. 81 f8 -> TS\_SECURITY\_HEADER::flagsHi - ignored as flags field does not contain SEC\_FLAGSHI\_VALID (0x8000)
18. 16 ff 53 cd 83 24 e0 da -> TS\_SECURITY\_HEADER1::dataSignature
19. ad 70 d2 f8 00 9d 97 29 71 0a 0e 7c cc 88 dd ac
20. b7 53 e0 0f 3e 83 -> Encrypted TS\_OFFSCRCACHE\_ERROR\_PDU
21. Decrypted TS\_OFFSCRCACHE\_ERROR\_PDU:
22. 00000000 16 00 17 00 ef 03 ea 03 01 00 00 01 c0 c0 2e 00 ................
23. 00000010 00 00 01 00 00 00 ......
24. 16 00 -> TS\_SHARECONTROLHEADER::totalLength = 0x0016 = 22 bytes
25. 17 00 -> TS\_SHARECONTROLHEADER::pduType = 0x0017
26. 0x0017
27. = 0x0010 | 0x0007
28. = TS\_PROTOCOL\_VERSION | PDUTYPE\_DATAPDU
29. ef 03 -> TS\_SHARECONTROLHEADER::pduSource = 0x03ef = 1007
30. ea 03 01 00 -> TS\_SHAREDATAHEADER::shareID = 0x000103ea
31. 00 -> TS\_SHAREDATAHEADER::pad1
32. 01 -> TS\_SHAREDATAHEADER::streamId = STREAM\_LOW (1)
33. c0 c0 -> TS\_SHAREDATAHEADER::uncompressedLength = 0xc0c0 (uninitialized due to bug)
34. 2e -> TS\_SHAREDATAHEADER::pduType2 = TS\_PDUTYPE2\_OFFSCRCACHE\_ERROR\_PDU (46)
35. 00 -> TS\_SHAREDATAHEADER::generalCompressedType = 0
36. 00 00 -> TS\_SHAREDATAHEADER::generalCompressedLength = 0
37. 01 00 00 00 -> TS\_OFFSCRCACHE\_ERROR\_PDU::flags = 0x00000001 = OC\_ERR\_FLUSH\_AND\_DISABLE\_OFFSCREEN

### DrawNineGrid Cache Error PDU

The following is an annotated dump of a [DrawNineGrid Cache Error PDU (section 2.2.2.3.3)](#Section_E871FE47A339464696C555605067B260).

1. 00000000 03 00 00 30 02 f0 80 64 00 06 03 eb 70 22 28 00 ...0...d....p"(.
2. 00000010 81 f8 2c 1e 5b bf a9 74 5f ee 08 63 79 ee 8a 51 ..,.[..t\_..cy..Q
3. 00000020 9c 08 80 2d 35 76 35 09 98 ff 69 78 cc 23 8f a4 ...-5v5...ix.#..
4. 03 00 00 30 -> TPKT Header (length = 48 bytes)
5. 02 f0 80 -> X.224 Data TPDU
6. 64 00 06 03 eb 70 22 -> PER encoded (ALIGNED variant of BASIC-PER) SendDataRequest
7. initiator = 1007 (0x03ef)
8. channelId = 1003 (0x03eb)
9. dataPriority = high
10. segmentation = begin | end
11. userData length = 0x22 = 34 bytes
12. 28 00 -> TS\_SECURITY\_HEADER::flags = 0x0028
13. 0x0028
14. = 0x0020 | 0x0008
15. = SEC\_IGNORE\_SEQNO | SEC\_ENCRYPT
16. 81 f8 -> TS\_SECURITY\_HEADER::flagsHi - ignored as flags field does not contain SEC\_FLAGSHI\_VALID (0x8000)
17. 2c 1e 5b bf a9 74 5f ee -> TS\_SECURITY\_HEADER1::dataSignature
18. 08 63 79 ee 8a 51 9c 08 80 2d 35 76 35 09 98 ff
19. 69 78 cc 23 8f a4 -> Encrypted TS\_DRAWNINEGRID\_ERROR\_PDU
20. Decrypted TS\_DRAWNINEGRID\_ERROR\_PDU:
21. 00000000 16 00 17 00 ef 03 ea 03 01 00 00 01 c0 c0 30 00 ..............0.
22. 00000010 00 00 01 00 00 00 ......
23. 16 00 -> TS\_SHARECONTROLHEADER::totalLength = 0x0016 = 22 bytes
24. 17 00 -> TS\_SHARECONTROLHEADER::pduType = 0x0017
25. 0x0017
26. = 0x0010 | 0x0007
27. = TS\_PROTOCOL\_VERSION | PDUTYPE\_DATAPDU
28. ef 03 -> TS\_SHARECONTROLHEADER::pduSource = 0x03ef = 1007
29. ea 03 01 00 -> TS\_SHAREDATAHEADER::shareID = 0x000103ea
30. 00 -> TS\_SHAREDATAHEADER::pad1
31. 01 -> TS\_SHAREDATAHEADER::streamId = STREAM\_LOW (1)
32. c0 c0 -> TS\_SHAREDATAHEADER::uncompressedLength = 0xc0c0 (uninitialized due to bug)
33. 30 -> TS\_SHAREDATAHEADER::pduType2 = TS\_PDUTYPE2\_DRAWNINEGRID\_ERROR\_PDU (48)
34. 00 -> TS\_SHAREDATAHEADER::generalCompressedType = 0
35. 00 00 -> TS\_SHAREDATAHEADER::generalCompressedLength = 0
36. 01 00 00 00 -> TS\_DRAWNINEGRID\_ERROR\_PDU::flags = 0x00000001 = DNG\_ERR\_FLUSH\_AND\_DISABLE\_DRAWNINEGRID

### GDI+ Error PDU

The following is an annotated dump of a [GDI+ Error PDU (section 2.2.2.3.4)](#Section_E370124B1E9A4A079FC4DB2A08A1E180).

1. 00000000 03 00 00 30 02 f0 80 64 00 06 03 eb 70 22 28 00 ...0...d....p"(.
2. 00000010 81 f8 44 c9 aa df dd e9 2e 50 cb ad d8 79 99 5f ..D......P...y.\_
3. 00000020 64 e2 e0 34 9f 33 a7 9b 53 0b 5f bc 40 4b 55 3b d..4.3..S.\_.@KU;
5. 03 00 00 30 -> TPKT Header (length = 48 bytes)
6. 02 f0 80 -> X.224 Data TPDU
7. 64 00 06 03 eb 70 22 -> PER encoded (ALIGNED variant of BASIC-PER) SendDataRequest
8. initiator = 1007 (0x03ef)
9. channelId = 1003 (0x03eb)
10. dataPriority = high
11. segmentation = begin | end
12. userData length = 0x22 = 34 bytes
13. 28 00 -> TS\_SECURITY\_HEADER::flags = 0x0028
14. 0x0028
15. = 0x0020 | 0x0008
16. = SEC\_IGNORE\_SEQNO | SEC\_ENCRYPT
17. 81 f8 -> TS\_SECURITY\_HEADER::flagsHi - ignored as flags field does not contain SEC\_FLAGSHI\_VALID (0x8000)
18. 44 c9 aa df dd e9 2e 50 -> TS\_SECURITY\_HEADER1::dataSignature
19. cb ad d8 79 99 5f 64 e2 e0 34 9f 33 a7 9b 53 0b
20. 5f bc 40 4b 55 3b -> Encrypted TS\_DRAWGDIPLUS\_ERROR\_PDU
21. Decrypted TS\_DRAWGDIPLUS\_ERROR\_PDU:
22. 00000000 16 00 17 00 ef 03 ea 03 01 00 00 01 c0 c0 31 00 ..............1.
23. 00000010 00 00 01 00 00 00 ......
24. 16 00 -> TS\_SHARECONTROLHEADER::totalLength = 0x0016 = 22 bytes
25. 17 00 -> TS\_SHARECONTROLHEADER::pduType = 0x0017
26. 0x0017
27. = 0x0010 | 0x0007
28. = TS\_PROTOCOL\_VERSION | PDUTYPE\_DATAPDU
29. ef 03 -> TS\_SHARECONTROLHEADER::pduSource = 0x03ef = 1007
30. ea 03 01 00 -> TS\_SHAREDATAHEADER::shareID = 0x000103ea
31. 00 -> TS\_SHAREDATAHEADER::pad1
32. 01 -> TS\_SHAREDATAHEADER::streamId = STREAM\_LOW (1)
33. c0 c0 -> TS\_SHAREDATAHEADER::uncompressedLength = 0xc0c0 (unitialized due to bug)
34. 31 -> TS\_SHAREDATAHEADER::pduType2 = TS\_PDUTYPE2\_DRAWGDIPLUS\_ERROR\_PDU (49)
35. 00 -> TS\_SHAREDATAHEADER::generalCompressedType = 0
36. 00 00 -> TS\_SHAREDATAHEADER::generalCompressedLength = 0
37. 01 00 00 00 -> TS\_DRAWGDIPLUS\_ERROR\_PDU::flags = 0x00000001 = GDIPLUS\_ERR\_FLUSH\_AND\_DISABLE\_DRAWGDIPLUS

# Security

## Security Considerations for Implementers

There are no security considerations for Remote Desktop Protocol: Graphics Devices Interfaces (GDI) Acceleration Extension as all traffic is secured by the underlying RDP core protocol. For an overview of the implemented security-related mechanisms, see [[MS-RDPBCGR]](file:///E:\Target\Windows\Published\Books\MS-RDPEGDI\%5bMS-RDPBCGR%5d.pdf) section 5.

## Index of Security Parameters

None.

# Appendix A: Product Behavior

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

* Windows NT operating system
* Windows 2000 operating system
* Windows XP operating system
* Windows Server 2003 operating system
* Windows Vista operating system
* Windows Server 2008 operating system
* Windows 7 operating system
* Windows Server 2008 R2 operating system
* Windows Server 2008 R2 operating system with Service Pack 1 (SP1)
* Windows 7 operating system with Service Pack 1 (SP1)
* Windows 8 operating system
* Windows Server 2012 operating system
* Windows 8.1 operating system
* Windows Server 2012 R2 operating system
* Windows 10 operating system
* Windows Server 2016 Technical Preview operating system

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

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

[<1> Section 2.2.2.2.1.1.1.4](#Appendix_A_Target_1): The Windows implementation of Remote Desktop Protocol: GDI Acceleration Extension places the field that contains the number of Delta-Encoded Points immediately before order encoding packets containing a [DELTA\_PTS\_FIELD](#Section_6c7b2a52103c4a7da2a9997416d4a475) structure.

[<2> Section 2.2.2.2.1.1.2.15](#Appendix_A_Target_2): This Unicode value is only for debugging purposes and is not always accurate in Microsoft RDP implementations.

[<3> Section 2.2.2.2.1.1.2.18](#Appendix_A_Target_3): The Windows implementation of RDP set this field to 0x0000.

[<4> Section 2.2.2.2.1.2.7](#Appendix_A_Target_4): The Windows implementation of Remote Desktop Protocol: Graphics Devices Interfaces (GDI) Acceleration Extension sets this field to 0x00.

# Change Tracking

This section identifies changes that were made to this document since 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](mailto:dochelp@microsoft.com).

| Section | Tracking number (if applicable) and description | Major change (Y or N) | Change type |
| --- | --- | --- | --- |
| [4.1.15](#Section_f7503c12d82d4c7f822c0b39d879f31b) FastGlyph | Updated the FastGlyph code example regarding using the encode flags in OpTop. | N | Content update. |
| [6](#Section_1be95eff27d04db0ba85c40cceeaf114) Appendix A: Product Behavior | Added Windows 10 to applicability list. | Y | Content update. |

# Index

**A**

Abstract data model

client ([section 3.1.1](#section_025e393573584f33a624207075a86c79) 122, [section 3.2.1](#section_f9cb0f79606640468f4f79b14b583bfb) 160)

[RDP 6.0-based bulk data compression](#section_08c389bf1a3c40e8af0f6e32654fd8f8) 126

server ([section 3.1.1](#section_025e393573584f33a624207075a86c79) 122, [section 3.3.1](#section_323a308d96ef4b4f97115a3e6cb31154) 172)

[Accelerated graphics](#section_f58e3854b38d46b8be59a7cd28725f1a) 22

[caches](#section_a922b0f77dc94b558567be4d1038d97f) 14

drawing orders ([section 1.3.1.2](#section_5b9fca3d644c4938af359c54e709f1d3) 14, [section 2.2.2.1.1](#section_762f3dd36a76485786a4a3ca951ca3a6) 23)

error conditions ([section 1.3.1.3](#section_fb96c39d167a4f40a77344bec1c47df8) 16, [section 2.2.2.3](#section_7b5607a14b3d44a5a2ef2ff36928248c) 109)

[overview](#section_804c39483cd04169ac343778b672137b) 13

Alternate secondary drawing orders ([section 1.3.1.2.3](#section_6255b5ecf77e49f0a4bf77b6c748cd64) 16, [section 2.2.2.2.1.3](#section_b69a751e34df4326b4e10e582ff7ea97) 96, [section 3.2.5.1.3](#section_26d5dda321c441de9ae6a0965ffdefe8) 169, [section 3.3.5.1.3](#section_4a200c71754048e5948a3fb120fd3ed9) 182)

[ALTSEC\_DRAWING\_ORDER\_HEADER packet](#section_b93614406e084a0697e76e1b7a4cc204) 96

[Annotated alternate secondary drawing orders examples](#section_d06ac0421d0c4d9e993cb0fdaa2e67b0) 223

[Annotated primary drawing orders examples](#section_14a7a47f48dc4dc488280595c0c45f6a) 186

[Annotated secondary drawing orders examples](#section_e363f0fae13640f48f1d3299680e1c44) 211

[Applicability](#section_785a3d772a174ac692698f1abe3f6bb8) 17

**B**

Bitmap

[compressing](#section_749fa4d40d884fe5bf4c914cd05776cb) 158

[decompressing](#section_872615ffe1ac469b94482c4452b0d21b) 159

[Save Bitmap](#section_c464f205346c475fb3d228fcef86a5ad) 161

[Bitmap Cache Wait List](#section_fb7b28f140c94535bd60b4fd345d94c8) 173

[Bitmap caches](#section_2bf9258842bd45278b3eb90c56e292d2) 122

[Bitmap keys - persistent](#section_4f9f384944f947f0aab6112d28ca6197) 172

[Brush caches](#section_a7e8809dd24343b5be0b25f8fd827f61) 124

**C**

[CACHE\_BITMAP\_ORDER packet](#section_c9365b320be547c9af1afeba2ea1ee9f) 80

[CACHE\_BITMAP\_REV2\_ORDER packet](#section_3c76f8fbfcaa4ca69d2e3f449bdbed40) 81

[CACHE\_BITMAP\_REV3\_ORDER packet](#section_710cf5c26a0c4cea8d62fa4f639d7c5a) 94

[CACHE\_BRUSH\_ORDER packet](#section_9e7027c4d9894cb09674e2e36c95ebdd) 88

[CACHE\_COLOR\_TABLE\_ORDER packet](#section_49fad30181714f38a68a7394baeb7b15) 84

[CACHE\_GLYPH\_ORDER packet](#section_6dc06088b1124f4fb0bcb97bba621a0c) 85

[CACHE\_GLYPH\_REV2\_ORDER packet](#section_ffc80435136e442e908540ca1d27a413) 86

[Caches](#section_a922b0f77dc94b558567be4d1038d97f) 14

[bitmap caches](#section_2bf9258842bd45278b3eb90c56e292d2) 122

[brush caches](#section_a7e8809dd24343b5be0b25f8fd827f61) 124

[color table caches](#section_b31800c6da774acc854bb954ebc1848b) 124

[fragment caches](#section_6d4a8cc0d4814766b609778b09c1ceac) 123

[GDI+ caches](#section_819a758e7ba24e6f96058b7b006cdad3) 125

[glyph caches](#section_6d4a8cc0d4814766b609778b09c1ceac) 123

[NineGrid bitmap caches](#section_b71d528445c146e7b92511edd3b1de82) 125

[offscreen bitmap caches](#section_63061daa980843c9b00dd5eda7dc8c21) 124

[overview](#section_b1c96aad5c7245969324619a0a6a31ae) 122

[Capability negotiation](#section_ea09f786f83e4aedae81c999068d0c95) 17

[Capability sets](#section_e14afe8cdf1a4488a18a75d1055da14f) 18

[Change tracking](#section_6faadbdbab6b4072ae9f111ba78a0a12) 255

Client

abstract data model ([section 3.1.1](#section_025e393573584f33a624207075a86c79) 122, [section 3.2.1](#section_f9cb0f79606640468f4f79b14b583bfb) 160)

[alternate secondary drawing orders](#section_26d5dda321c441de9ae6a0965ffdefe8) 169

[drawing orders](#section_2ec8788d47b24188ba5483d5f9a6efb9) 162

higher-layer triggered events ([section 3.1.4](#section_9740a20b629d44eea9c4d21737e0b154) 125, [section 3.2.4](#section_c654614b7006431985d1cdfb41533642) 161)

initialization ([section 3.1.3](#section_588141349f0a4efc934b24a3248a82ab) 125, [section 3.2.3](#section_f283162747894576b4ef779f0c641740) 161)

local events ([section 3.1.7](#section_f78e1b4eeb244fbb8db1bf6bc483932c) 126, [section 3.2.7](#section_5c45e59ea9494d5481dbbffe3f122837) 172)

message processing ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.2.5](#section_4cbdeb652bc7428fa9afdecc916dee21) 162)

[other local events](#section_5c45e59ea9494d5481dbbffe3f122837) 172

[primary drawing orders](#section_90a5d432d45f4b4b96b3cdc3775e85f2) 162

[RDP 6.0 bitmap compression](#section_0d2e1a03e12346b2b2b8ed730a794ae4) 148

[RDP 6.0-based bulk data compression](#section_1bfeaea6bba74d09893bb47f127c94d5) 126

[secondary drawing orders](#section_a2c4470e0dd84caf976774dae9c1750a) 167

sequencing rules ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.2.5](#section_4cbdeb652bc7428fa9afdecc916dee21) 162)

timer events ([section 3.1.6](#section_898bc01c8af34aeca54e210182d52a9b) 125, [section 3.2.6](#section_a11ce699238245b0a96eead311f72255) 172)

timers ([section 3.1.2](#section_794750f2ed3942e6a5691e8f5eaeaf4c) 125, [section 3.2.2](#section_09c3f50e5b6b4e47a41ced8e6f62b26b) 161)

[Client\_Bitmap\_Cache\_Error\_PDU packet](#section_fdc28a58010549caa9040fe3bee58ec2) 109

[Client\_DrawNineGrid\_Cache\_Error\_PDU packet](#section_e871fe47a339464696c555605067b260) 113

[Client\_GDIplus\_Error\_PDU packet](#section_e370124b1e9a4a079fc4db2a08a1e180) 115

[Color table caches](#section_b31800c6da774acc854bb954ebc1848b) 124

[Compressed stream - decoding](#section_f652cccf9be24f14a025d3247efb7f5f) 136

[COMPRESSED\_COLOR\_BRUSH packet](#section_43f2b0a56a7647a3b6d279b08b80f2d2) 90

[COORD\_FIELD packet](#section_870b982c9abe40cb99d02c3f0cc5fb74) 24

[Copy-offset tables](#section_0fffd61e94064a85bc77c712093bc95e) 127

[CREATE\_NINEGRID\_BITMAP\_ORDER packet](#section_6ac5eb820c30405490e883797f24bc38) 99

[CREATE\_OFFSCR\_BITMAP\_ORDER packet](#section_427758ff4bac4833b388d634cc512d0c) 97

**D**

Data

[compressing - RDP 6.0-based bulk data compression](#section_506a6581d43e440c98b29a48bd9a4974) 126

[decompressing - RDP 6.0-based bulk data compression](#section_2757a6b238954bd39b14decbadd89809) 126

Data model - abstract

client ([section 3.1.1](#section_025e393573584f33a624207075a86c79) 122, [section 3.2.1](#section_f9cb0f79606640468f4f79b14b583bfb) 160)

[RDP 6.0-based bulk data compression](#section_08c389bf1a3c40e8af0f6e32654fd8f8) 126

server ([section 3.1.1](#section_025e393573584f33a624207075a86c79) 122, [section 3.3.1](#section_323a308d96ef4b4f97115a3e6cb31154) 172)

[DELTA\_PTS\_FIELD packet](#section_6c7b2a52103c4a7da2a9997416d4a475) 25

[DELTA\_RECTS\_FIELD packet](#section_b89f2058b1804da09bd1aa694c87768c) 26

[DRAW\_GDIPLUS\_CACHE\_END\_ORDER packet](#section_b44bd2b06c00454db182dbb841131c7a) 106

[DRAW\_GDIPLUS\_CACHE\_FIRST\_ORDER packet](#section_53dbc9c54e3f4b17a256baf43c582146) 104

[DRAW\_GDIPLUS\_CACHE\_NEXT\_ORDER packet](#section_20da0b5e8cc94805998e03e9f216bade) 105

[DRAW\_GDIPLUS\_CACHE\_TYPE packet](#section_9dee1f7b3989470dba973d78f4111099) 103

[DRAW\_GDIPLUS\_END\_ORDER packet](#section_657e77cd01d94eccaf45a69e0b48390f) 108

[DRAW\_GDIPLUS\_FIRST\_ORDER packet](#section_03941c5e98d64f61ac74ddb307f1f317) 106

[DRAW\_GDIPLUS\_NEXT\_ORDER packet](#section_b5deca1d5802428998cb87035e1ca454) 107

Drawing orders

alternate secondary ([section 1.3.1.2.3](#section_6255b5ecf77e49f0a4bf77b6c748cd64) 16, [section 2.2.2.2.1.3](#section_b69a751e34df4326b4e10e582ff7ea97) 96)

alternate secondary drawing orders ([section 3.2.5.1.3](#section_26d5dda321c441de9ae6a0965ffdefe8) 169, [section 3.3.5.1.3](#section_4a200c71754048e5948a3fb120fd3ed9) 182)

[client](#section_2ec8788d47b24188ba5483d5f9a6efb9) 162

overview ([section 1.3.1.2](#section_5b9fca3d644c4938af359c54e709f1d3) 14, [section 2.2.2.1.1](#section_762f3dd36a76485786a4a3ca951ca3a6) 23)

primary ([section 1.3.1.2.1](#section_a8a67fe14117453d9ea2ab67bc1c10ef) 15, [section 2.2.2.2.1.1](#section_b0d203b0b32d4b0d84e5bc21eb11a3ec) 24)

primary drawing orders ([section 3.2.5.1.1](#section_90a5d432d45f4b4b96b3cdc3775e85f2) 162, [section 3.3.5.1.1](#section_ca34709d88dd481cbc816f9ffb5c82a2) 174)

secondary ([section 1.3.1.2.2](#section_f9b049d67dcf4d1686349c82143dff9e) 15, [section 2.2.2.2.1.2](#section_9df0ec924b494302aa863421151b0c11) 77)

secondary drawing orders ([section 3.2.5.1.2](#section_a2c4470e0dd84caf976774dae9c1750a) 167, [section 3.3.5.1.2](#section_576991c0ca0e4be4a2bbf0a6e7948d67) 179)

[server](#section_def5fa73043548dabc7f8d4c0484d1e9) 173

[DRAWING\_ORDER packet](#section_5a67b33f446d41059355fe5de076975c) 24

[DRAWNINEGRID\_ORDER packet](#section_44bd6a68a5b74a38980706ba4ab4a72a) 75

[DSTBLT\_ORDER packet](#section_87ea30df59d6438ea73583f0225fbf91) 47

**E**

[ELLIPSE\_CB\_ORDER packet](#section_897203838394423a810aba631d66055f) 74

[ELLIPSE\_SC\_ORDER packet](#section_5146fab3c43f44d7b54104666a80e33b) 73

[EOS tables](#section_0fffd61e94064a85bc77c712093bc95e) 127

Error conditions ([section 1.3.1.3](#section_fb96c39d167a4f40a77344bec1c47df8) 16, [section 2.2.2.3](#section_7b5607a14b3d44a5a2ef2ff36928248c) 109, [section 3.2.5.2](#section_13c3c84e5f1e4ac1a562619c2815fbd8) 171, [section 3.3.5.2](#section_390feb125e8947b593f065a2fb6a8063) 184)

[Examples](#section_b39b1eb5f1bc445786b2080b95c655fa) 186

[annotated alternate secondary drawing orders examples](#section_d06ac0421d0c4d9e993cb0fdaa2e67b0) 223

[annotated primary drawing orders examples](#section_14a7a47f48dc4dc488280595c0c45f6a) 186

[annotated secondary drawing orders examples](#section_e363f0fae13640f48f1d3299680e1c44) 211

[glyph image example](#section_05d60ffebe3f4026878cd22ce2b6fb03) 247

[NineGrid example](#section_06ba35beca674fd1937e7a0ac678b63a) 243

[SaveBitmap example](#section_5337b5e437084570a6821791ffc90c88) 245

**F**

[FASTGLYPH\_ORDER packet](#section_839e963ce24a42adb4bdeb108340629c) 68

[FASTINDEX\_ORDER packet](#section_ac55f919acb943c1a3ec33802929f9f7) 67

[Fields - vendor-extensible](#section_88c9a7787e8641cda11e68c74eaf694b) 17

[FILL\_MODE packet](#section_06dc707471cd41cf9028964877875dfc) 43

[FOUR\_BYTE\_UNSIGNED\_ENCODING packet](#section_f6f16e2b876742ee8eff547780cca952) 79

[Fragment caches](#section_6d4a8cc0d4814766b609778b09c1ceac) 123

[FRAME\_MARKER packet](#section_9ba0e3d204ae4c73a35ecb854f587018) 109

**G**

[GDI+ caches](#section_819a758e7ba24e6f96058b7b006cdad3) 125

[Glossary](#section_fd46ff0b6bf84aefa0b34ccaf72356b3) 11

[Glyph caches](#section_6d4a8cc0d4814766b609778b09c1ceac) 123

[Glyph image example](#section_05d60ffebe3f4026878cd22ce2b6fb03) 247

[GLYPHINDEX\_ORDER packet](#section_8e906465a1da4d0691db54beb7827381) 63

**H**

Higher-layer triggered events

client ([section 3.1.4](#section_9740a20b629d44eea9c4d21737e0b154) 125, [section 3.2.4](#section_c654614b7006431985d1cdfb41533642) 161)

server ([section 3.1.4](#section_9740a20b629d44eea9c4d21737e0b154) 125, [section 3.3.4](#section_4b9778c0795944c0a6fe867fbfeb09f9) 173)

**I**

[Implementer - security considerations](#section_958bc24dec5a47b2addb8e32b13e9222) 253

[Index of security parameters](#section_11a739e717fe484b94b659df2f72dc90) 253

[Informative references](#section_0424bc8cbcbc4a528fa434f4345ed2c0) 13

Initialization

client ([section 3.1.3](#section_588141349f0a4efc934b24a3248a82ab) 125, [section 3.2.3](#section_f283162747894576b4ef779f0c641740) 161)

server ([section 3.1.3](#section_588141349f0a4efc934b24a3248a82ab) 125, [section 3.3.3](#section_30cb8aa6a44a494d8d37e389326b86cb) 173)

[Introduction](#section_3042756207b44b69abb9a6a3916364c6) 11

**L**

[Length-of-match tables](#section_21aebf8fb46c43488518bf444a3d5c16) 131

[LINETO\_ORDER packet](#section_27276fcf32be49a99a1797ec6167ac7e) 61

[Literal tables](#section_0fffd61e94064a85bc77c712093bc95e) 127

Local events

client ([section 3.1.7](#section_f78e1b4eeb244fbb8db1bf6bc483932c) 126, [section 3.2.7](#section_5c45e59ea9494d5481dbbffe3f122837) 172)

server ([section 3.1.7](#section_f78e1b4eeb244fbb8db1bf6bc483932c) 126, [section 3.3.7](#section_dca68cff42b748ff89e204b542fb5885) 185)

[Logically-compressed stream - encoding](#section_fb37ff5718264811892b42266fdd5301) 133

**M**

[MEM3BLT\_ORDER packet](#section_3aa21a4a031c4a3f8fc5159068f30237) 59

[MEMBLT\_ORDER packet](#section_84c2ec2ff776405b9b486894a28b1b14) 57

Message processing

client ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.2.5](#section_4cbdeb652bc7428fa9afdecc916dee21) 162)

server ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.3.5](#section_2686d734c19b4bccb65ea433afcd72e6) 173)

Messages

[syntax](#section_6049130c5e5b4cd9b4765fc6a39169a2) 18

[transport](#section_381c8b7073994a928e3241ce51c5a6b1) 18

[MULTI\_DRAWNINEGRID\_ORDER packet](#section_101fecdc3ad841c6b7b670d8efd4ad70) 76

[MULTI\_DSTBLT\_ORDER packet](#section_aaae78403eb440389204b95f87b44534) 48

[MULTI\_OPAQUERECT\_ORDER packet](#section_74e46cd9d6494bbe9f8af4200f23302e) 55

[MULTI\_PATBLT\_ORDER packet](#section_5f677576a62e48d9a58d711490882365) 52

[MULTI\_SCRBLT\_ORDER packet](#section_31b79b4915b74709bbc2d58ce0362292) 56

**N**

[NineGrid bitmap caches](#section_b71d528445c146e7b92511edd3b1de82) 125

[NineGrid example](#section_06ba35beca674fd1937e7a0ac678b63a) 243

[NINEGRID\_BITMAP\_INFO packet](#section_ba44b18260ad4bb199d55d00c0695bcb) 100

[Normative references](#section_b71b07c08a1b45498c1fc14a647a5738) 12

**O**

[OFFSCR\_DELETE\_LIST packet](#section_e40b1297f5ad46c8a83ee7529f1d5c5b) 98

[Offscreen bitmap caches](#section_63061daa980843c9b00dd5eda7dc8c21) 124

[Offscreen\_Bitmap\_Cache\_Error\_PDU packet](#section_c78f94f32a7f41a8902a1c6b30d34240) 111

[OPAQUERECT\_ORDER packet](#section_1eead7aaac63411a9f8cb1b227526877) 54

Other local events

[client](#section_5c45e59ea9494d5481dbbffe3f122837) 172

[server](#section_dca68cff42b748ff89e204b542fb5885) 185

[Overview (synopsis)](#section_f7cffa8e54c346339ea88f09e1347c60) 13

**P**

[Parameters - security index](#section_11a739e717fe484b94b659df2f72dc90) 253

[PATBLT\_ORDER packet](#section_bd4bf5e7b98845f982013b22cc9aeeb8) 49

[Persistent bitmap keys](#section_4f9f384944f947f0aab6112d28ca6197) 172

[POLYGON\_CB\_ORDER packet](#section_24f4e2291f9f4633b9be66d7579ba36d) 71

[POLYGON\_SC\_ORDER packet](#section_22e1117fd40d41df970959389ac1f8dc) 70

[POLYLINE\_ORDER packet](#section_b60b4d9baac54b9884d3edc0c0779a5c) 72

[Preconditions](#section_f3fed10a522a430e96f0ea000662e71e) 17

[Prerequisites](#section_f3fed10a522a430e96f0ea000662e71e) 17

Primary drawing order history ([section 3.2.1.1](#section_f1da06b585c940c7a0de3f280f480960) 161, [section 3.3.1.2](#section_19b1b7f2456c4d928f5a5f507947889c) 172)

Primary drawing orders ([section 1.3.1.2.1](#section_a8a67fe14117453d9ea2ab67bc1c10ef) 15, [section 2.2.2.2.1.1](#section_b0d203b0b32d4b0d84e5bc21eb11a3ec) 24, [section 3.2.5.1.1](#section_90a5d432d45f4b4b96b3cdc3775e85f2) 162, [section 3.3.5.1.1](#section_ca34709d88dd481cbc816f9ffb5c82a2) 174)

[PRIMARY\_DRAWING\_ORDER packet](#section_23f766d483434e6b8281071ddccc0272) 43

[Product behavior](#section_1be95eff27d04db0ba85c40cceeaf114) 254

**R**

RDP 6.0 bitmap compression

[client](#section_0d2e1a03e12346b2b2b8ed730a794ae4) 148

[compressing bitmaps](#section_749fa4d40d884fe5bf4c914cd05776cb) 158

[decompressing bitmaps](#section_872615ffe1ac469b94482c4452b0d21b) 159

[run-length encoding](#section_c5f101a07e674f3ca678d9760ce8921c) 152

[server](#section_0d2e1a03e12346b2b2b8ed730a794ae4) 148

[techniques](#section_770e00520ebc4d13b3267c7338595c67) 148

[RDP 6.0 bulk compression](#section_2baf663546d64f32982400c41d9b5c81) 16

RDP 6.0-based bulk data compression

[abstract data model](#section_08c389bf1a3c40e8af0f6e32654fd8f8) 126

[client](#section_1bfeaea6bba74d09893bb47f127c94d5) 126

[compressing data](#section_506a6581d43e440c98b29a48bd9a4974) 126

[decompressing data](#section_2757a6b238954bd39b14decbadd89809) 126

[server](#section_1bfeaea6bba74d09893bb47f127c94d5) 126

[wire format](#section_c3ce7d8919244b58a8c8623440571be0) 127

[RDP6\_BITMAP\_STREAM packet](#section_9b422f698e054c6db6fbfa02ef75a8f2) 118

[RDP6\_RLE\_SEGMENT packet](#section_f7e4c717669e4f318b34e8f5ab2e107e) 120

[RDP6\_RLE\_SEGMENTS packet](#section_623172526dff4b5682909e8b5513665b) 120

[RDP61\_COMPRESSED\_DATA packet](#section_4b2427d8f80048229a3d82cf1ccd3ce7) 116

[RDP61\_MATCH\_DETAILS packet](#section_01f927654cd844bf95f212b91d127dd5) 118

[References](#section_d217355fa65c45b0b00f6e7f659ae375) 12

[informative](#section_0424bc8cbcbc4a528fa434f4345ed2c0) 13

[normative](#section_b71b07c08a1b45498c1fc14a647a5738) 12

[Relationship to other protocols](#section_641d5df9e0044b408e5ac0d94cbed79d) 17

[ROP2\_OPERATION packet](#section_34e554e42e7c4f1fa8a6e7b319bf6fe1) 27

[ROP3\_OPERATION\_INDEX packet](#section_a9a85075e79645ebb84af399324a1109) 28

[Run-length encoding](#section_c5f101a07e674f3ca678d9760ce8921c) 152

**S**

[Save Bitmap](#section_c464f205346c475fb3d228fcef86a5ad) 161

[SaveBitmap example](#section_5337b5e437084570a6821791ffc90c88) 245

[SAVEBITMAP\_ORDER packet](#section_db4ce8dbcf76483d99ba354cd010ea4a) 62

[SCRBLT\_ORDER packet](#section_a4e322b0cd644dfc8e1af24dc0edc99d) 55

Secondary drawing orders ([section 1.3.1.2.2](#section_f9b049d67dcf4d1686349c82143dff9e) 15, [section 2.2.2.2.1.2](#section_9df0ec924b494302aa863421151b0c11) 77, [section 3.2.5.1.2](#section_a2c4470e0dd84caf976774dae9c1750a) 167, [section 3.3.5.1.2](#section_576991c0ca0e4be4a2bbf0a6e7948d67) 179)

[SECONDARY\_DRAWING\_ORDER\_HEADER packet](#section_c54e2667335d4a59a6e27dc9744dbe79) 77

[Security](#section_3d4a86f7d3b64573a2b8965448f2ac06) 253

[implementer considerations](#section_958bc24dec5a47b2addb8e32b13e9222) 253

[parameter index](#section_11a739e717fe484b94b659df2f72dc90) 253

Sequencing rules

client ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.2.5](#section_4cbdeb652bc7428fa9afdecc916dee21) 162)

server ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.3.5](#section_2686d734c19b4bccb65ea433afcd72e6) 173)

Server

abstract data model ([section 3.1.1](#section_025e393573584f33a624207075a86c79) 122, [section 3.3.1](#section_323a308d96ef4b4f97115a3e6cb31154) 172)

[alternate secondary drawing orders](#section_4a200c71754048e5948a3fb120fd3ed9) 182

[drawing orders](#section_def5fa73043548dabc7f8d4c0484d1e9) 173

higher-layer triggered events ([section 3.1.4](#section_9740a20b629d44eea9c4d21737e0b154) 125, [section 3.3.4](#section_4b9778c0795944c0a6fe867fbfeb09f9) 173)

initialization ([section 3.1.3](#section_588141349f0a4efc934b24a3248a82ab) 125, [section 3.3.3](#section_30cb8aa6a44a494d8d37e389326b86cb) 173)

local events ([section 3.1.7](#section_f78e1b4eeb244fbb8db1bf6bc483932c) 126, [section 3.3.7](#section_dca68cff42b748ff89e204b542fb5885) 185)

message processing ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.3.5](#section_2686d734c19b4bccb65ea433afcd72e6) 173)

[other local events](#section_dca68cff42b748ff89e204b542fb5885) 185

[primary drawing orders](#section_ca34709d88dd481cbc816f9ffb5c82a2) 174

[RDP 6.0 bitmap compression](#section_0d2e1a03e12346b2b2b8ed730a794ae4) 148

[RDP 6.0-based bulk data compression](#section_1bfeaea6bba74d09893bb47f127c94d5) 126

[secondary drawing orders](#section_576991c0ca0e4be4a2bbf0a6e7948d67) 179

sequencing rules ([section 3.1.5](#section_611ecc04b9e94f46bd87d0e57ee7d3d4) 125, [section 3.3.5](#section_2686d734c19b4bccb65ea433afcd72e6) 173)

timer events ([section 3.1.6](#section_898bc01c8af34aeca54e210182d52a9b) 125, [section 3.3.6](#section_eaa9ef00c79b47daa971fe517be6bdfa) 185)

timers ([section 3.1.2](#section_794750f2ed3942e6a5691e8f5eaeaf4c) 125, [section 3.3.2](#section_43391aa8ecbe4bc994e15eeae50c471e) 173)

[Standards assignments](#section_51f4271f37aa42c1a4a48c525cb997bb) 17

[STREAM\_BITMAP\_FIRST\_ORDER packet](#section_b714a7c97fd54621a397349a4ef3e06c) 101

[STREAM\_BITMAP\_NEXT\_ORDER packet](#section_dfe099ffa894406393a7fbe64e7cb85e) 102

[SWITCH\_SURFACE\_ORDER packet](#section_ada4d8a9421e48c0a0ed7384fa3cf061) 98

[Syntax - message](#section_6049130c5e5b4cd9b4765fc6a39169a2) 18

**T**

Timer events

client ([section 3.1.6](#section_898bc01c8af34aeca54e210182d52a9b) 125, [section 3.2.6](#section_a11ce699238245b0a96eead311f72255) 172)

server ([section 3.1.6](#section_898bc01c8af34aeca54e210182d52a9b) 125, [section 3.3.6](#section_eaa9ef00c79b47daa971fe517be6bdfa) 185)

Timers

client ([section 3.1.2](#section_794750f2ed3942e6a5691e8f5eaeaf4c) 125, [section 3.2.2](#section_09c3f50e5b6b4e47a41ced8e6f62b26b) 161)

server ([section 3.1.2](#section_794750f2ed3942e6a5691e8f5eaeaf4c) 125, [section 3.3.2](#section_43391aa8ecbe4bc994e15eeae50c471e) 173)

[Tracking changes](#section_6faadbdbab6b4072ae9f111ba78a0a12) 255

[Transport](#section_381c8b7073994a928e3241ce51c5a6b1) 18

[Transport - message](#section_381c8b7073994a928e3241ce51c5a6b1) 18

Triggered events - higher-layer

client ([section 3.1.4](#section_9740a20b629d44eea9c4d21737e0b154) 125, [section 3.2.4](#section_c654614b7006431985d1cdfb41533642) 161)

server ([section 3.1.4](#section_9740a20b629d44eea9c4d21737e0b154) 125, [section 3.3.4](#section_4b9778c0795944c0a6fe867fbfeb09f9) 173)

[TS\_BITMAP\_CACHE\_ERROR\_INFO packet](#section_2ecfb696fb2b4aca9202612828adade6) 111

[TS\_BITMAP\_CACHE\_ERROR\_PDU packet](#section_484c7e89fc5f4f14b2d1f3d019d8a6b6) 110

[TS\_CACHE\_GLYPH\_DATA packet](#section_faf6e70bfd0f4905801061c4439fbcad) 86

[TS\_CACHE\_GLYPH\_DATA\_REV2 packet](#section_7facfe557db14a47a38ebc223cce5dc4) 87

[TS\_COLOR packet](#section_f99c616b876d437199fecc656d0a610b) 42

[TS\_COLOR\_QUAD packet](#section_982ce4ebff014eeba5049ba1ee4a722b) 84

[TS\_COLORREF packet](#section_d8a54901807c4b3fb7c9d50cf01b62ce) 101

[TS\_COLORTABLE\_CAPABILITYSET packet](#section_2b7c69463612429195a803b7b1387eaf) 18

[TS\_DRAW\_GDIPLUS\_CAPABILITYSET packet](#section_52635737d1444f479c88b48ceaf3efb4) 19

[TS\_DRAW\_NINEGRID\_CAPABILITYSET packet](#section_c7fff28863db4521bbe577e060fb0780) 18

[TS\_DRAWGDIPLUS\_ERROR\_PDU packet](#section_43c4679794654d21aeb2a1dcd79578b3) 116

[TS\_DRAWNINEGRID\_ERROR\_PDU packet](#section_d88a4151e1c04bb4a77e897bca29008b) 114

[TS\_FP\_UPDATE\_ORDERS packet](#section_7f53bb2892e94cbea15f8553c902e3c4) 23

[TS\_GDIPLUS\_CACHE\_CHUNK\_SIZE packet](#section_cfe6022291854ab4ab06b0ec6e2d174e) 21

[TS\_GDIPLUS\_CACHE\_ENTRIES packet](#section_f7328112a3a445ea9022a28033912273) 21

[TS\_GDIPLUS\_IMAGE\_CACHE\_PROPERTIES packet](#section_73c1767cf10e4361b980f39e3c44f895) 22

[TS\_OFFSCRCACHE\_ERROR\_PDU packet](#section_f12c4d8b533548b6933e807480b237ce) 112

[TS\_UPDATE\_ORDERS\_PDU\_DATA packet](#section_ef7c27f5097746fea50ef9a51d6ab90b) 22

[TWO\_BYTE\_SIGNED\_ENCODING packet](#section_7dcab5880143463fa11c63bbd6f30090) 78

[TWO\_BYTE\_UNSIGNED\_ENCODING packet](#section_7baeb5f283844d22b6475c66f05be4f7) 78

**V**

[VARIABLE1\_FIELD packet](#section_008cb2338ae94e05ad0383f284f20a61) 25

[VARIABLE2\_FIELD packet](#section_aead742ea85847f1a7790311660f84af) 25

[Vendor-extensible fields](#section_88c9a7787e8641cda11e68c74eaf694b) 17

[Versioning](#section_ea09f786f83e4aedae81c999068d0c95) 17

**W**

Wire format - RDP 6.0-based bulk data compression

[decoding compressed stream](#section_f652cccf9be24f14a025d3247efb7f5f) 136

[encoding logically compressed stream](#section_fb37ff5718264811892b42266fdd5301) 133

[length-of-match tables](#section_21aebf8fb46c43488518bf444a3d5c16) 131

[literal and EOS and copy-offset tables](#section_0fffd61e94064a85bc77c712093bc95e) 127

[overview](#section_c3ce7d8919244b58a8c8623440571be0) 127