QXDM Professional

Software User Guide

80-V1241-21 YE

October 9, 2007

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Contents

1 Introduction 14

1.1 Purpose 14
1.2 Scope 14
1.3 Conventions 14
1.4 Revision history 15
1.5 References 19
1.6 Technical assistance 19
1.7 Acronyms 19

2 Installation 20

2.1 Required hardware and software 20
2.1.1 Minimum system requirements 20
2.2 Installing QXDM 21
2.2.1 Installing from CD-ROM 21
2.2.2 Installing over the network inside QUALCOMM 22
2.2.3 Installing over the network outside QUALCOMM 22
2.2.4 QLMS activation 22
2.2.5 WIBU dongle usage 22
2.3 Physical connectivity 23
2.4 Overview 24
2.4.1 Installation path 24
2.4.2 QXDM Tools Suite 24
2.4.2.1 Database Editor 24
2.4.2.2 DLF Converter 24
2.4.2.3 ISF Converter 25
2.4.2.4 Item Tester 25
2.4.2.5 Listen-Only QXDM 25
2.4.2.6 QXDM 25
2.4.2.7 PPP Extractor 25
2.4.2.8 RPC proxy 26
2.4.3 QXDM documentation 26
2.4.3.1 QUALCOMM ISF Processing Interface 26
2.4.3.2 QXDM Database Editor User Guide 26
2.4.3.3 ReadMe 26
2.5 Item Store Format (.ISF) files 26
2.5.1 Always-On logging 27

2.5.2 Recovering the Item Store 27
2.5.3 Saving the Item Store 27

3 Display Overview 28

3.1 File menu 28
3.1.1 Annotate 28
3.1.2 Dynamic Parsers 29
3.1.3 View Registrations 30
3.1.3.1 Context options 31
3.1.4 Load Configuration… 31
3.1.5 Save Configuration… 32
3.1.6 New Items… 32
3.1.7 Load Items… 33
3.1.8 Save Items… 34
3.1.9 Replay Items… 35
3.1.9.1 Maximum Gap 35
3.1.9.2 Playback Speed 35
3.1.9.3 Compact Layout 35
3.1.9.4 Step 35
3.1.9.5 Play/Pause 35
3.1.10 Item Store Settings… 36
3.1.10.1 General Options 36
3.1.10.2 Mode Options 37
3.1.11 Exit 38
3.2 View menu 39
3.2.1 New 39
3.2.2 Command Bar 39
3.2.3 Status Bar 39
3.2.4 View Bar 39
3.3 Options menu 40
3.3.1 Communications… 40
3.3.1.1 Target Port 41
3.3.1.2 GPS Server Port 41
3.3.1.3 Timeouts (MS) 42
3.3.2 Settings… 43
3.3.2.1 Font Settings 44
3.3.2.2 Alternate Script Path 44
3.3.2.3 Build Source Path 44
3.3.2.4 Vendor Database Path 45
3.3.2.5 WCDMA Protocol Revision 45
3.3.2.6 CDMA protocol revision 45
3.3.2.7 HSDPA UE Category 45
3.3.3 DIP Switches… 46
3.3.4 Log View Config… 46
3.3.5 Message View Config… 47
3.3.6 Parsing Options 47

3.3.6.1 Export ANSI Text 47
3.3.6.2 Export Parsed Text 47
3.3.6.3 Parsing DLLs, Default DB, User DB 48
3.3.6.4 Parsing DLLs, User DB, Default DB 48
3.3.6.5 Default DB, User DB, Parsing DLLs 48
3.3.6.6 User DB, Default DB, Parsing DLLs 48
3.3.7 Auto-Launch Core Dump View 48
3.3.8 Hide Hex Pane By Default 48
3.3.9 Track Dropped Messages 48
3.4 Tools menu 49
3.5 Window menu 49
3.5.1 Tag Window 49
3.5.2 Clear Most 50
3.5.3 Clear All 50
3.5.4 Close 50
3.5.5 Close All 50
3.5.6 Arrange Icons 50
3.5.7 Cascade 50
3.5.8 Minimize All 50
3.5.9 Restore All 50
3.5.10 Tile Horizontal/Tile Vertical 50
3.5.11 Previous Window/Next Window 51
3.5.12 Active Window List 51
3.6 Help menu 51
3.6.1 Licensing 51
3.7 View Bar 52
3.8 Command Bar 52
3.8.1 Command interface 52
3.8.1.1 Using filenames 52
3.8.2 Script Help for command interface 53
3.9 Status Bar 53
3.9.1 Config… 53

4 QXDM Views 55

4.1 QXDM views 55
4.2 Item list views 56
4.2.1 Scrolling List Pane 57
4.2.1.1 Menu options 57
4.2.1.2 Appearance 58
4.2.1.3 Configuration 59
4.2.1.4 Copying 60
4.2.1.5 Searching 62
4.2.1.6 Auto-Scroll 63
4.2.1.7 Clear Items 63
4.2.1.8 Match Items… 64
4.2.1.9 Process Items 65

4.2.1.10 Refilter Items 67
4.2.1.11 Raw Item 68
4.2.1.12 Sync Near Item 69
4.2.1.13 Sync To Item 70
4.2.2 Raw Item Pane 71
4.2.2.1 Menu options 71
4.2.3 Parsed Item Pane 72
4.2.3.1 Menu options 72
4.3 HTML views 73
4.3.1 Menu options 74
4.4 Graph views 75
4.4.1 Clear 77
4.4.2 Cursor 77
4.4.3 Save Image 77
4.4.4 Auto-Scroll 77
4.4.5 Axis Scroll Mode 77
4.4.6 Axis Zoom Mode 77
4.4.7 Legend Visible 77
4.4.8 Range 77
4.4.9 View Plots 78
4.4.10 View Channel 79
4.5 Other views 80
4.5.1 Application Statistics 80
4.5.1.1 Communications Port 80
4.5.1.2 Item Totals 80
4.5.1.3 Item Store 80
4.5.2 Dynamic Item View 81
4.5.3 NV Browser 83
4.5.3.1 Offline 84
4.5.3.2 Reset 84
4.5.3.3 Read 84
4.5.3.4 Write 84
4.5.4 Keypad 84
4.5.5 Memory Viewer <F4> 85
4.5.5.1 Address 85
4.5.5.2 Rows 86
4.5.5.3 Write button 86
4.5.5.4 Context menu 86
4.5.6 OS Core Dump 86
4.5.7 Task Profiling 87
4.5.8 Debug Trace View 90
4.5.9 CDMA Finger Placement 92
4.5.9.1 Finger statistics 92
4.5.9.2 Delay (chips) 92
4.5.9.3 Current E_c/I_0 (dB) 93
4.5.9.4 E_c/I_0 (dB) 93
4.5.10 Status 93

5 Log View 94
5.1 Log View 94
5.2 Log View Configuration 94
5.2.1 Log Packets 95
5.2.2 Message Packets 96
5.2.3 Event Reports 97
5.2.4 Diagnostic Logs 98
5.2.4.1 GSM NV Items 99
5.2.4.2 IS-2000 Items 99
5.2.4.3 PC Polling Time 99
5.2.5 Strings 100
5.2.6 Misc 101

6 Messages View 102
6.1 Messages View 102
6.2 Message View Configuration 103
6.2.1 Log Packets 103
6.2.2 Log Packets (OTA) 104
6.2.3 Message Packets 105
6.2.4 Event Reports 106
6.2.5 Strings 107

7 Command Prompt Interface 108
7.1 QXDM properties 108
7.2 Command functions 109
7.2.1 CWait 110
7.2.2 Echo 110
7.2.3 Logging 111
7.2.4 LogMask 111
7.2.5 Mode 112
7.2.6 Pause 112
7.2.7 RequestItem 113
7.2.8 RequestItemFile 113
7.2.9 RequestNamedItem 114
7.2.10 RequestNamedItemFile 115
7.2.11 RequestNVItemRead 116
7.2.12 RequestNVItemWrite 116
7.2.13 RequestNVItemIDRead 117
7.2.14 RequestNVItemIDWrite 117
7.2.15 Run 118
7.2.16 SendRawRequest 118
7.2.17 Wait 118
7.2.18 WaitForItem 119

8 COM Automation Interfaces 120
8.1 Overview 120
8.2 QXDM Interface (IQXDM) 121
8.2.1 AppVersion 121
8.2.2 ClearViewItems 121
8.2.3 COMPort 121
8.2.4 CopyViewItems 122
8.2.5 CloseView 122
8.2.6 CreateView 123
8.2.7 DipSwitchMask 123
8.2.8 ExportViewText 124
8.2.9 GetIQXDM2 124
8.2.10 GPSPort 124
8.2.11 IsPhoneConnected 124
8.2.12 LoadConfig 125
8.2.13 LoadItemStore 125
8.2.14 LogMask 125
8.2.15 LogMaskOff 125
8.2.16 LogMaskOn 126
8.2.17 OfflineDigital 126
8.2.18 ResetPhone 126
8.2.19 SaveConfig 126
8.2.20 SendDmIcdPacket 126
8.2.21 SendDmIcdPacketEx 127
8.2.22 SendScript 127
8.2.23 SetCDMAProtocolRevision 128
8.2.24 SetWCDMAProtocolRevision 128
8.2.25 SaveItemStore 129
8.2.26 SetLoggingOn 129
8.2.27 SetLoggingOnEx 129
8.2.28 SetLoggingOff 130
8.2.29 SetVendorDatabase 130
8.2.30 QuitApplication 130
8.2.31 QXDMTextOut 130
8.2.32 Visible 131
8.2.33 WaitEvent 131
8.3 Client Interface (IQXDM2) 132
8.3.1 RequestItem 132
8.3.2 RemoveRequest 133
8.3.3 GetServerState 133
8.3.4 GetItemCount 134
8.3.5 GetItem 134
8.3.6 RegisterClient 134
8.3.7 RegisterQueueClient 135

8.3.8 UnregisterClient 135
8.3.9 ConfigureClientByKeys 135
8.3.10 ConfigureClientByNames 136
8.3.11 ClearClientItems 136
8.3.12 GetClientItemCount 137
8.3.13 GetClientItem 137
8.3.14 ClientRequestItem 138
8.3.15 ClientRemoveRequest 139
8.3.16 ClientRequestNVRead 140
8.3.17 ClientRequestNVWrite 140
8.3.18 CopyClientItems 141
8.3.19 SyncToItem 141
8.3.20 SyncToClientItem 142
8.4 Client Config Interface (IClientConfig) 142
8.4.1 ClearConfig 142
8.4.2 CommitConfig 142
8.4.3 AddItem 143
8.4.4 AddDIAGRequest 144
8.4.5 AddDIAGResponse 145
8.4.6 AddSubsysRequest 146
8.4.7 AddSubsysResponse 147
8.4.8 AddEvent 148
8.4.9 AddLog 149
8.4.10 AddMessage 150
8.4.11 AddString 151
8.4.12 AddOTALog 152
8.4.13 AddOTALogByString 154
8.4.14 SetSubsysV2DelayedResponsesOnly 154
8.5 Item Interface (IColorItem) 155
8.5.1 GetItemType 155
8.5.2 GetItemTypeText 155
8.5.3 GetItemColor 155
8.5.4 GetItemTimestamp 156
8.5.5 GetItemTimestamp2 156
8.5.6 GetItemTimestampText 156
8.5.7 GetItemSpecificTimestamp 156
8.5.8 GetItemSpecificTimestamp2 157
8.5.9 GetItemSpecificTimestampText 157
8.5.10 GetItemBuffer 157
8.5.11 GetItemDLFBuffer 158
8.5.12 GetItemBufferText 158
8.5.13 GetItemKeyText 158
8.5.14 GetItemName 158
8.5.15 GetItemSummary 159

8.5.16 GetItemSize 159
8.5.17 GetItemParsedText 159
8.5.18 GetItemFieldValue 160
8.5.19 GetItemFieldValueText 161
8.5.20 GetItemFields 161
8.5.21 GetNamedItemFields 161
8.5.22 GetConfiguredItemFields 162
8.6 Field Interface (IColorItemFields) 163
8.6.1 GetXML 163
8.6.2 GetFieldCount 163
8.6.3 GetFieldIndex 164
8.6.4 GetFieldIndexFrom 164
8.6.5 GetFieldIndexByID 165
8.6.6 GetFieldIndexFromByID 165
8.6.7 GetFieldOffset 166
8.6.8 GetFieldSize 166
8.6.9 GetFieldValue 167
8.6.10 GetFieldName 168
8.6.11 GetFieldValueText 168
8.7 QXDM automation instances 169
8.7.1 Launching QXDM from a remote computer 170

9 Runtime Parsing DLLs 174

10 Frequently Asked Questions 175

Figures

Figure 2-1 Physical connectivity 23
Figure 2-2 Start Menu – QXDM Tools Suite 24
Figure 3-1 File menu options 28
Figure 3-2 File → Dynamic Parsers… 29
Figure 3-3 File → View Registrations… 30
Figure 3-4 File → Load Configuration… 31
Figure 3-5 File → Save Configuration… 32
Figure 3-6 File → Load Items… 33
Figure 3-7 File → Save Items… 34
Figure 3-8 Replay Items 35
Figure 3-9 Item Store File Settings 36
Figure 3-10 View menu 39
Figure 3-11 Options menu 40
Figure 3-12 Options → Communications… dialog 40
Figure 3-13 QPST configuration globe 41
Figure 3-14 QPST Port Server configuration 41
Figure 3-15 Options → Settings… dialog 43
Figure 3-16 Options → DIP Switches… dialog 46
Figure 3-17 Parsing Options 47
Figure 3-18 Tools menu 49
Figure 3-19 Window menu 49
Figure 3-20 Help menu 51
Figure 3-21 View Bar 52
Figure 3-22 Command Bar 52
Figure 3-23 Status Bar 53
Figure 3-24 Status Bar Config 53
Figure 4-1 QXDM views using the View bar 55
Figure 4-2 Context-sensitive menu for Scrolling List Pane 57
Figure 4-3 Item List Appearance 58
Figure 4-4 Item List Config dialog 59
Figure 4-5 Copy To Items File Progress dialog 61
Figure 4-6 Find option for scrolling views 62
Figure 4-7 Match Items 64
Figure 4-8 Process With Script 66
Figure 4-9 Refilter Items 67
Figure 4-10 Raw Item menu 68
Figure 4-11 Sync Near Item 69
Figure 4-12 Sync To Item 70
Figure 4-13 Raw Item Pane menu 71
Figure 4-14 Parsed Item Pane menu 72
Figure 4-15 HTML view menu 74
Figure 4-16 Graph view context menus 76
Figure 4-17 View Plots context menu option 78
Figure 4-18 View Channel context menu option 79

Figure 4-19 Dynamic Item Config 81
Figure 4-20 Dynamic Item Views 82
Figure 4-21 NV Browser 83
Figure 4-22 Memory Viewer context menu 85
Figure 4-23 Task List Pane 87
Figure 4-24 Task Profiling Configuration 88
Figure 4-25 Debug Trace View dump files 90
Figure 4-26 Debug Trace Dump retrieved items 91
Figure 4-27 CDMA Finger Placement 92
Figure 5-1 Options → Log View Configuration → Log Packets 95
Figure 5-2 Options → Log View Configuration → Message Packets 96
Figure 5-3 Options → Log View Configuration → Event Reports 97
Figure 5-4 Options → Log View Configuration → Diagnostic Logs 98
Figure 5-5 Options → Log View Configuration → Strings 100
Figure 5-6 Options → Log View Configuration → Misc 101
Figure 6-1 Options → Message View Configuration → Log Packets 103
Figure 6-2 Options → Message View Configuration → Log Packets (OTA) 104
Figure 6-3 Options → Message View Configuration → Message Packets 105
Figure 6-4 Options → Message View Configuration → Event Reports 106
Figure 6-5 Options → Message View Configuration → String 107
Figure 7-1 Item Tester QXDM Command Example 109
Figure 7-2 Copy from Item Tester application to QXDM command 110
Figure 8-1 Distributed COM configuration properties (Windows XP) 170
Figure 8-2 QXDM location configuration 171
Figure 8-3 QXDM security configuration 172
Figure 8-4 QXDM Identity configuration 173

Tables

Table 1-1 Revision history 15
Table 1-2 Reference documents and standards 19
Table 2-1 Minimum system requirements 20
Table 2-2 PPP Extractor operation 25
Table 3-1 Status Bar Indicators 54
Table 4-1 Item types 60
Table 4-2 Copying Items 60
Table 4-3 ProcessItem() parameters 65
Table 4-4 Graph views 75
Table 4-5 Task Profiling Task Fields 89
Table 7-1 Property command replacements 108
Table 7-2 CWait parameters 110
Table 7-3 Echo parameters 110
Table 7-4 Logging parameters 111
Table 7-5 LogMask parameters 111
Table 7-6 Mode parameters 112
Table 7-7 RequestItem parameters 113
Table 7-8 RequestItemFile parameters 113
Table 7-9 RequestNamedItem parameters 114
Table 7-10 RequestNamedItemFile parameters 115

Table 7-11 RequestNVItemRead parameters 116
Table 7-12 RequestNVItemWrite parameters 116
Table 7-13 RequestNVItemIDRead parameters 117
Table 7-14 RequestNVItemIDWrite parameters 117
Table 7-15 Run parameters 118
Table 7-16 SendRawRequest parameters 118
Table 7-17 Wait 118
Table 7-18 WaitForItem parameters 119
Table 8-1 ClearViewItems parameter 121
Table 8-2 CopyViewItems parameters 122
Table 8-3 CloseView parameters 122
Table 8-4 CreateView parameters 123
Table 8-5 DipSwitchMask parameter 123
Table 8-6 ExportViewText parameters 124
Table 8-7 LoadConfig parameter 125
Table 8-8 LoadItemStore parameters 125
Table 8-9 LogMaskOff parameters 125
Table 8-10 LogMaskOn parameters 126
Table 8-11 SaveConfig parameters 126
Table 8-12 SendDmIcdPacketEx parameters 127
Table 8-13 SendScript parameter 127
Table 8-14 SetCDMAProtocolRevision parameters 128
Table 8-15 SetWCDMAProtocolRevision parameters 128
Table 8-16 SaveItemStore parameter 129
Table 8-17 SetLoggingOnEx parameter 129
Table 8-18 SetLoggingOff parameter 130
Table 8-19 SetVendorDatabase parameters 130
Table 8-20 QXDMTextOut parameter 130
Table 8-21 Visible parameter 131
Table 8-22 WaitEvent parameters 131
Table 8-23 RequestItem parameters 132
Table 8-24 RemoveRequest parameters 133
Table 8-25 GetServerState return values 133
Table 8-26 GetItem parameters 134
Table 8-27 RegisterClient parameters 134
Table 8-28 RegisterQueueClient parameters 135
Table 8-29 UnregisterClient parameters 135
Table 8-30 ConfigureClientByKeys parameters 135
Table 8-31 ConfigureClientByNames parameters 136
Table 8-32 ClearClientItems parameters 136
Table 8-33 GetClientItemCount parameters 137
Table 8-34 GetClientItem parameters 137
Table 8-35 ClientRequestItem parameters 138
Table 8-36 ClientRemoveRequest parameters 139
Table 8-37 ClientRequestNVRead parameters 140
Table 8-38 ClientRequestNVRead parameters 140
Table 8-39 CopyClientItems parameters 141
Table 8-40 SyncToItem parameters 141
Table 8-41 SyncToClientItem parameters 142
Table 8-42 AddItem parameters 143
Table 8-43 AddDIAGRequest parameters 144

Table 8-44 AddDIAGResponse parameters 145
Table 8-45 AddSubsysRequest parameters 146
Table 8-46 AddSubsysResponse parameters 147
Table 8-47 AddEvent parameters 148
Table 8-48 AddLog parameters 149
Table 8-49 AddLog parameters 150
Table 8-50 AddString parameters 151
Table 8-51 AddOTALog parameters 152
Table 8-52 OTA log codes/keys (CDMA) 152
Table 8-53 OTA log codes/keys (CDMA 1xEV-DO) 153
Table 8-54 OTA log codes/keys (GSM) 153
Table 8-55 OTA log codes/keys (UMTS) 153
Table 8-56 OTA log codes/keys (WCDMA) 153
Table 8-57 OTA log codes/keys (Bluetooth) 153
Table 8-58 AddOTALogByString parameters 154
Table 8-59 SetSubsysV2DelayedResponsesOnly 154
Table 8-60 GetItemType return values 155
Table 8-61 GetItemTimestampText parameters 156
Table 8-62 GetItemSpecificTimestampText parameters 157
Table 8-63 GetItemBuffer parameters 157
Table 8-64 GetItemBufferText parameters 158
Table 8-65 GetItemSize parameters 159
Table 8-66 GetItemFieldValue parameters 160
Table 8-67 Supported field types 160
Table 8-68 GetItemFieldValueText parameters 161
Table 8-69 GetNamedItemFields parameters 161
Table 8-70 GetConfiguredItemFields parameters 162
Table 8-71 GetFieldIndex parameters 164
Table 8-72 GetFieldIndexFrom parameters 164
Table 8-73 GetFieldIndexByID parameters 165
Table 8-74 GetFieldIndexFromByID parameters 165
Table 8-75 GetFieldOffset parameters 166
Table 8-76 GetFieldSize parameters 166
Table 8-77 GetFieldValue parameters 167
Table 8-78 Supported field types 167
Table 8-79 GetFieldName parameters 168
Table 8-80 GetFieldValueText parameters 168
Table 10-1 User color item keys 176

1 Introduction

1.1 Purpose

The QUALCOMM® Extensible Diagnostic Monitor (QXDM) provides a diagnostic client for Dual-Mode Subscriber Station (DMSS) and newer User Equipment (UE) software, Advanced Mobile Subscriber Software (AMSS).

QXDM was developed to provide a rapid prototyping platform for new diagnostic clients and diagnostic protocol packets. It provides a Graphical User Interface (GUI) that displays data transmitted to and from the DMSS.

NOTE The use of DMSS in this document refers loosely to user equipment that is connected to QXDM using the QUALCOMM diagnostic interface.

1.2 Scope

This user guide is intended for users of QXDM who need to know how to install, use, and understand the information provided by QXDM. Many features for specific diagnostic clients supported by QXDM are documented elsewhere. Applicable documents are referenced in Table 1-2. Additionally, features added since this revision, are documented in the Release Notes for the software release in which the feature was added and can be referenced via the Start → Programs → QXDM → ReadMe file.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Key and/or button names appear in a different font, e.g., press OK, or click EDIT.

Keys that are pressed in combination are indicated with a plus sign, e.g., press CTRL + C.

Code variables appear in angle brackets, e.g., `<number>`.

Shading indicates content that has been added or changed in this revision of the document.

1.4 Revision history

The revision history for this document is shown in Table 1-1.

Table 1-1 Revision history

Document number	Date	Description
80-V1241-1 Rev. –	Mar 2000	Initial release
80-V1241-3 Rev. –	Apr 2000	Added chapters on application screens, troubleshooting; added status bar description **Note**: There was no 80-V1241-2. The number was skipped intentionally in accordance with internal administrative tracking practices.
80-V1241-4 Rev. –	Apr 2000	Added temporal analyzer information; updated application screens chapter
80-V1241-9 Rev. –	Jun 2000	Added GPS feature; added sections and screens – RLP throughput display, quick paging channel, RLP3 statistics, NV items; updated chapters: installation, logging and events, application screens. **Note**: There was no 80-V1241-5 through 80-V1241-8. The numbers were skipped intentionally in accordance with internal administrative tracking practices.
80-V1241-14 Rev. –	Aug 2000	■ Added sections and screens – IS-2000 supplemental channel MUX parameters, retrievable parameters, streaming configuration, simple test data services, gpsOne® ■ Updated chapters – Overview of screens, application screens ■ Updated screens – Three splitter screen, NV items **Note**: There was no 80-V1241-10 through 80-V1241-13. The numbers were skipped intentionally in accordance with internal administrative tracking practices.
80-V1241-15 Rev. –	Sep 2000	■ Added sections and screens – Pilot sets display, temporal analyzer configuration, log priorities, GPS statistics, factory Test mode ■ Updated properties chapter; updated sections and/or screens: fast forward power control, IS-2000 supplemental channel MUX parameters, retrievable parameters, simple test data services, gpsOne; updated scope and intended audience, reference documents ■ Removed streaming configuration section
80-V1241-17 Rev. –	Nov 2000	■ Renamed overview of screens chapter to display overview chapter; added sections to the chapter: options (with screens), property views ■ Added sections and screens – Full test data services, events, Bluetooth™ logs ■ Updated logging and events chapter and screen; updated sections: property support tools, gpsOne, GPS statistics; updated screen – Simple test data services; updated reference documents **Note**: There was no 80-V1241-16. The number was skipped intentionally in accordance with internal administrative tracking practices.
80-V1241-18 Rev. –	Dec 2000	Added gpsOne reference document; updated sections and/or screen – gpsOne, GPS statistics; updated reference documents.
80-V1241-19 Rev. –	Feb 2001	Added property editor chapter; updated properties chapter; updated reference documents

Document number	Date	Description
80-V1241-20 Rev. –	Mar 2001	■ Added chapters – Events, QXDM command prompt functions ■ Updated Perl script interface chapter ■ Updated screens – MUX traffic channel statistics, MUX traffic channel and secondary statistics, quick paging channel statistics, fast forward power control logs, RLP3 statistics log, NV items, IS-2000 SCH MUX parameters, retrievable parameters, BT logs, Bluetooth logs
80-V1241-21 1	Jun 2001	■ Updated Sections 4.1-Options, 4.4.3-Messages view and autoscroll button, and 7.2.1-Scrolling display ■ Updated screens – RLP3 Statistics Log, IS-2000 SCH MUX Parameters, Bluetooth Logs, Full Test Data Services, gpsOne, Simple Test Data Services ■ Added screens Active Set Information and the following HDR screens – Air Link Summary, Fingers Data, Forward Link Statistics, GEN TA, Reverse Link Statistics, RLP Statistics, Tx Statistics, SEARCH Status, Status, Temporal Analyzer, and the following WCDMA screens – AGC, Block Error Rate, Downlink TM Channel Parameters, Layer 1 State, Layer 4 Connection Mgmt, MAC Channel Mapping, MAC Parameters, Mobility Management, Physical Channel, RLC DL UM Channel Parameters, RLC UL UM Channel Parameters, RRC Status, Temporal Analyzer, TFCS Downlink, Transport Channels, Uplink TM Channel Parameters **Note**: The document number for this revision has been changed to adhere to current document numbering standards.
80-V1241-21 2	Sep 2001	■ Rewrote Chapter 11 – Script Interface ■ Updated Annotate Logfile documentation, Section 4.1; updated ALIEM documentation, Section 4.2 ■ Updated Chapter 14 screens – Full Test Data Service, Simple Test Data Service ■ Removed Chapter 14 Sections – IS-2000 SCH MUX Parameters, WCDMA L4 Connection Management ■ Added Chapter 14 Sections – MUX Statistics, WCDMA CS and PS Connection Management, WCDMA NAS Error
80-V1241-21 A	Nov 2001	■ Rewrote Chapter 14 – Application Screens ■ Updated Property Support Tools, Sections 6.6 and 6.6.1 ■ Added documentation to Property Database, Sections 8.1 and 8.2 ■ Updated document version numbers and dates in Section 1.5 ■ The revision numbering system has been changed for internal tracking purposes only
B	Feb 2002	■ Removed Chapters 9 and 10 ■ Added a Serial Mode Preference section to the Communications dialog in Chapter 3 ■ Added PPP Logging to the Options menu ■ Added some properties, functions and examples to the Script Interface chapter ■ Numerous changes/additions were made to Table 13-1 in the Application Screens chapter
C	Mar 2002	■ For the Options dialog – Updated screen shots, and added a section about Messages ■ Added information to the Command Output view Section 4.4.1 ■ Updated screen shots and added information in Chapter 7 – Events ■ Added some methods and the descriptions for some properties and methods to the Script Interface chapter ■ Several changes/additions were made to Table 13-1 in the Application Screens chapter

Document number	Date	Description
D	Apr 2002	■ In Section 4.4.3 – Updated and added screen shots, and added information describing some significant modifications to the <F3>Messages display; some functionality of File and Options menus is now incorporated into the <F3> display ■ In Section 1.6, added a reminder to read the Readme file
E	Aug 2002	Updated chapters and Sections 1.5 References, 1.7 Acronyms, 2 Installation, 3 Communication Parameters, 4 Display Overview, 5 Logging, 6 Properties, 7 Events, 8 Properties, 10 QXDM Commands, 12.1 Application Screens, and 13 Troubleshooting
F	Sep 2002	■ Updated chapters 4 Display Overview, 5 Logging, 7 Events, 12 Application Screens ■ Removed Sections 7.3.1 and 7.3.2
G	Nov 2002	■ Updated Chapter 8 for the new Property Editor ■ Added WCDMA CM GSM Measurements display to Table 12–1 ■ Added Common Channel Stats display to Table 12–1 ■ Added splitter_view and send_data commands to Chapter 10, QXDM Command Prompt Functions ■ Updated Chapters 4 and 5 with the new Logging and Message View Configuration <Logging, F3 Messages, and F5/Ctrl+F5 configuration) ■ Updated Note 16 (12.1.16) for the WCMDA AGC display
H	May 2003	Updated the Installation, Display Overview, Messages View, Events, and Automation Interface chapters
J	Jun 2003	■ Replaced Property Editor chapter with Database Editor chapter ■ Added ComPort interface description to Automation Interface chapter **Note**: There is no Rev. I per Mil. standards.
K	Aug 2003	Added new command descriptions to the Automation Interface chapter
L	Dec 2003	Added new descriptions to the Communications Parameters, Display Overview, Logging, and Database Editor chapters; updated descriptions and figures for Messages View and Log View Configuration chapters
M	Apr 2004	Applied 3.5.00 (minor version) updates to Chapters 3-8, 11, and 13
N	Jun 2004	Update Windows® menu, Copying Items menu, and Automation Methods table; updated Sections 3.5, 4.1.1.3, Figures 3-17, 4-1, 4-3, 4-5, and Tables 4-2, 9-2.
P	Sep 2004	Added NV Browser and Dynamic Item View descriptions; updated descriptions for timeouts, context-sensitive help, launching remote sessions, runtime parsers, DB Editor, Properties, and Automation; changed Chapters 3, 4, 7-12, 14, Sections 3.3.1.3, 3.3.2, 3.3.2.3, 4.1.1, 4.1.1.10, 8.1.1, 8.3.1 to 8.3.5, 9.1.3, Figures 3-14, 4-1, 4-10, 4-13, 4-15, 8-5, 9-2, and Table 13-1. **Note**: There is no Rev. O per Mil. standards.
R	Nov 2004	Removed DB Editor chapter; updated COM Automation chapter with new COM interfaces; deprecated Properties chapter; updated Application Screens table. **Note**: There is no Rev. Q per Mil. standards.
T	Jan 2005	Updated Section 4.2 HTML views; added Section 8.2 QXDM interface (IQXDM); added Section 9.1 Database access functions; updated Chapter 11 FAQs; removed obsolete Application Screens chapter. **Note**: There is no Rev. S per Mil. standards.

Document number	Date	Description
U	Apr 2005	Updated File, View, and Options menu additions; Correct tables referencing item types; Add descriptions for REX Profiling, REX Core Dump, Debug Trace View, CDMA Finger Placement; Update description of graph views; Add Item Tester discussion for getting QXDM command examples; Remove obsolete Temporal Analyzer chapter; Add description for SaveItemStore automation function.
V	Jul 2005	Document Subsystem Dispatch V2 Delayed Response. Describe new automation functions GetConfiguredItemFields(), GetItemFieldValue(), GetItemFieldValueText(). Document new Options menu item Hide Hex Pane By Default. Document new context menu item Process Items… and Save Image…. Remove deprecated properties documentation. Update Memory Viewer description. Add descriptions for automation methods SyncToItem() and SyncToClientItem(), GetFieldIndex(), GetFieldIndexFrom(), GetFieldIndexByID(), GetFieldIndexFromByID(), GetItemSize().
W	Oct 2005	Combined chapters 7 and 9 into one chapter (7), Command Prompt Interface. Updated Sections 2.5, 3.1.1, 3.1.9, 3.1.10, 3.2, 3.3.4, 3.5, 3.8.1, 4.4 and, 4.5.2.
Y	Apr 2006	Updated Sections 2.4.1, 2.5.1, 3.3.6, 3.3.7, 3.3.9, 3.9.4, 5, 6 related to Always-On logging; added Sections 2.4.2 QXDM Tools Suite and 2.4.3 Documentation; Removed obsolete note in 8.2.23; Corrected swapped GPS Information/Target event item type descriptions. Added Section 2.2.4, Add RegisterQueueClient, GetItemDLFBuffer(), and floating point type descriptions. Updated Sections 2.5, 3.1.10, 3.3.2, 3.3.6, 3.3.9, 4.5, 4.6.2, 4.6.3, 5, 7.2.3, 7.2.4, 8.2.13, 8.2.23. **Note:** There is no Rev. X per Mil. standards.
YA	Jul 2006	Added GPSPort and SetSubsysV2DelayedResponsesOnly descriptions to Chapter 8. Added Sections 4.3.1.2.1, Delayed Subsystem Responses Only, and 4.3.1.2.2, Accept Unknowns. Added Parsing Options description. Updated description for Dynamic Item View. Corrected QLMS description. Replaced REX with OS/Task references for profiling and core dump views. Updated Section 2.1, Required Hardware and Software. Added Sections 3.3.2.3, Vendor Database Path, and 8.2.29, SetVendorDatabase. Updated Sections 2.5.1, Always-On logging, and 4.6.1.3, Item Store. **Note:** There is no Rev. Z per Mil. standards.
YB	Oct 2006	Updated file paths information for Windows Vista™ compliance. Updated Sections 3.1.10, Item Store Settings, 4.3.1.4.2, Find options. Removed references to legacy SIA feature (3.3.5, 3.9.2, Figure 3-11, Figure 3-17). Moved support information to [Q8]. Updated Section 3.9.1, ALIEM. Updated installation instructions. Removed license file section. Removed CFA sections.
YC	Oct 2006	Added Document View Registrations → Copy Text menu option. Updated view bar, command bar, and status bar sections. Rewrote most of view section.
YD	Nov 2006	Updated Section 2.4.2.7, described PPP Extractor. Updated the Preferences dialog documentation by adding Section 3.3.2.1 to reflect font support. Updated Section 4.3 to document the HTML view CSS file and the HTML view context menu operation.
YE	Oct 2007	Clarified installation location and definition of PC Time. Added dongle support documentation. Added HSDPA UE category documentation. Updated FAQ. Updated Help menu.

1.5 References

Reference documents, which may include QUALCOMM, standards, and resource documents, are listed in Table 1-2. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-2 Reference documents and standards

Ref.	Document	
QUALCOMM		
Q1	*CDMA DMSS Serial Data Interface Control Document*	80-V1294-1
Q2	*Application Note: Factory Test Mode*	CL93-V1974-1
Q3	*gpsOne® Position Determination Messaging and Parameters*	80-V0726-1
Q4	*Serial Interface Control Document for WCDMA*	80-V2708-1
Q5	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q6	*QXDM Database Editor User Guide*	80-V9396-1
Q7	*QUALCOMM® ISF Processing Interfaces Specification*	80-V5627-1
Q8	*Obtaining Support for QXDM*	80-VD384-1

NOTE Refer to Section 2.4.3 for access to QXDM documentation.

1.6 Technical assistance

For support information, refer to [Q8].

An additional important source of information is the readme file that can be accessed from your Start menu (after installation) by selecting Start → Programs → QXDM → Readme. This file contains a list of all changes and additions to QXDM for each release, and should be read upon each installation of a new version.

1.7 Acronyms

For definitions of terms and abbreviations, refer to [Q5].

2 Installation

2.1 Required hardware and software

QXDM is designed to be installed and run on a workstation running Microsoft® Windows® 2000, Service Pack 1 or higher.

NOTE QXDM requires Microsoft Internet Explorer® 6.0 and QPST™ Server Ver 2.7, Build 200 or higher (QPST 2.7.200). For convenience, QPST is included with QXDM on CD-ROM or can be installed separately over network installations.

2.1.1 Minimum system requirements

The minimum required configuration for QXDM is described in Table 2-1.

NOTE QXDM's extensibility features make it possible to support unlimited configurations including support for end user-defined applications. A minimum configuration necessarily means that less work can be done simultaneously. Multiple instances of QXDM, other running (including background) processes, and having too many views open at the same time may require more than the minimum system configuration listed below.

Table 2-1 Minimum system requirements

Item	Description
CPU	800 MHz Pentium III class
RAM	256 MB
Hard drive	Installation requires 30 MB. Logging requires min 20 MB free disk space. Note that some of today's phones are capable of generating log data in excess of 18 MB per minute.
Operating system	Windows XP, Windows 2000 Service Pack 1
Port connections	One USB or serial I/O port per device (phone, FFA, SURF™, or GPS receiver)

2.2 Installing QXDM

The QXDM software is provided either over the network or on CD-ROM.

The installer sets up the QXDM execution environment, which includes installing application binaries, data files, and documentation; registering COM automation components and file associations; and configuring QXDM for initial use.

The QXDM installation consists of two main folders.

- The QXDM program folder – The path to this folder is set by the user when installing QXDM. The default path offered by the installer is based on the underlying Microsoft® operating system program files folder which is typically C:\Program Files\Qualcomm\QXDM. After installation, this folder will contain subfolders containing the QXDM binaries, the QXDM documentation, and parsing DLL wizards for the Microsoft Visual Studio® development environment. The contents of this folder should be considered read-only.
- The QXDM data folder – The base path to this folder is set by the underlying Microsoft operating system and represents the documents folder shared by all users of the host PC. Typically the Microsoft Windows shared documents folder is located at C:\Documents and Settings\All Users\Documents. After installation, this folder will contain a Qualcomm\QXDM subfolder. The resulting complete path represents the QXDM data folder. Under this will be several subfolders containing QXDM automation script samples, the QXDM database, implementation files for all QXDM HTML-based displays, temporary QXDM item store format files, user submitted QXDM extensions, reference dynamic parsing DLLs, and the reference QUALCOMM vendor database. Unlike the program folder, the QXDM data folder is designed to accommodate user extensions, such as a user database or a user authored QXDM HTML display.

Two folders are utilized in order to be compliant with the current Microsoft Windows Logo Program requirements.

The installer creates a QXDM folder in the Windows Start Programs menu that can be run by selecting Start → All Programs → QXDM → QXDM. The installed application binaries and user guides are accessible from this location. Additionally a shortcut to the QXDM data folder is installed.

NOTE Attempting to install QXDM by running the QXDMInstaller.msi file is not supported. The only supported means of installing QXDM is via the setup.exe program.

2.2.1 Installing from CD-ROM

To install from a CD-ROM:

1. From the Windows desktop, select Start → Run → D:\Setup.exe. Specify the drive letter of your CD-ROM.
2. Press ENTER.

2.2.2 Installing over the network inside QUALCOMM

To install via the QUALCOMM network, utilize Advertised Programs. Assistance with Advertised Programs can be found on the QUALCOMM IT website.

QXDM engineering build releases are typically provided on a weekly basis. We recommend that you install the latest version of QXDM periodically to receive the most recent updates. Builds are announced to the email list *asw.crm.release*. Subscribe to this list to receive up-to-date notifications. This email list is also used to notify when builds are failed.

2.2.3 Installing over the network outside QUALCOMM

Follow the instructions provided by QUALCOMM for installing over the network.

2.2.4 QLMS activation

In some versions of QXDM, a one-time online activation is required before QXDM can be run using QUALCOMM License Management System (QLMS). Follow the instructions provided by QUALCOMM to activate your copy of QXDM.

2.2.5 WIBU dongle usage

NOTE This section was added to this document revision.

In some versions of QXDM, support for the WIBU dongle protection key system is available as a replacement for QLMS. WIBU dongle support requires the following:

- WIBU dongle runtime environment – The WIBU runtime environment can be installed by following the instructions located at http://wibu.com/download_user.php?lang=en
- QUALCOMM-programmed WIBU dongle protection key
- QUALCOMM License File (QLF) that matches the above WIBU dongle protection key installed to the QXDM data folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM)

2.3 Physical connectivity

QXDM connects to a phone or SURF through the QPST Server (Section 3.3.1.1) using a serial or USB cable to a COM port on your PC, as illustrated in Figure 2-1.

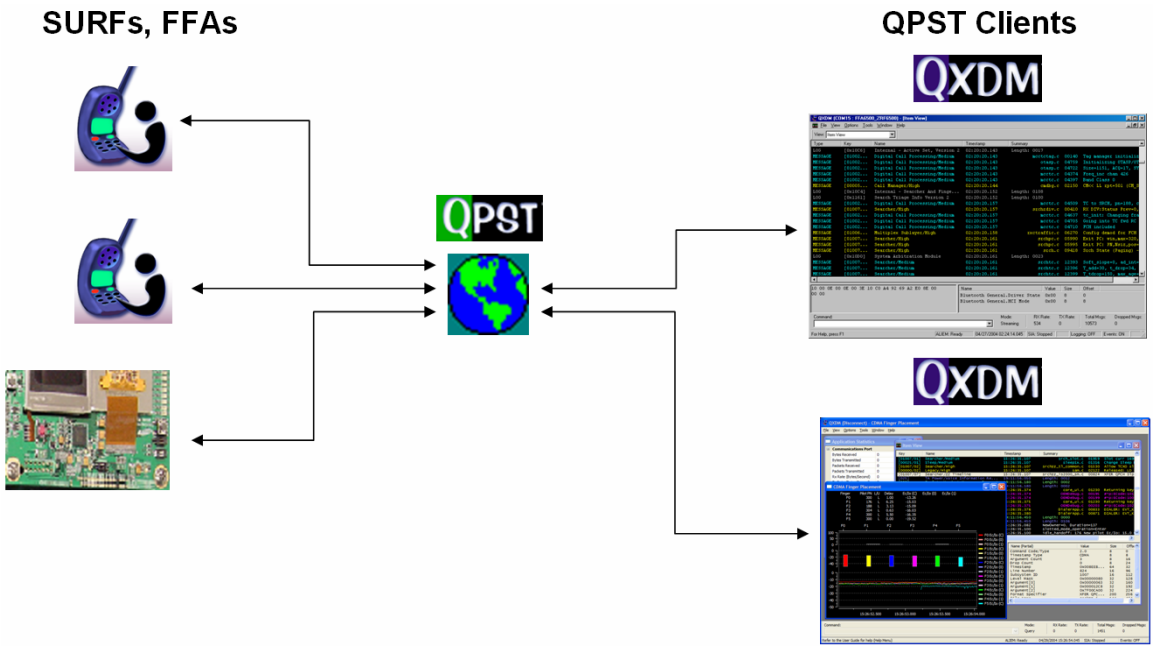

Figure 2-1 Physical connectivity

2.4 Overview

2.4.1 Installation path

The QXDM installation consists of two main folders. Refer to Section 2.2 for more information.

2.4.2 QXDM Tools Suite

The following tools are installed with QXDM and can be accessed from the Windows Start menu. More commonly used tools are also available through QXDM's Tools menu.

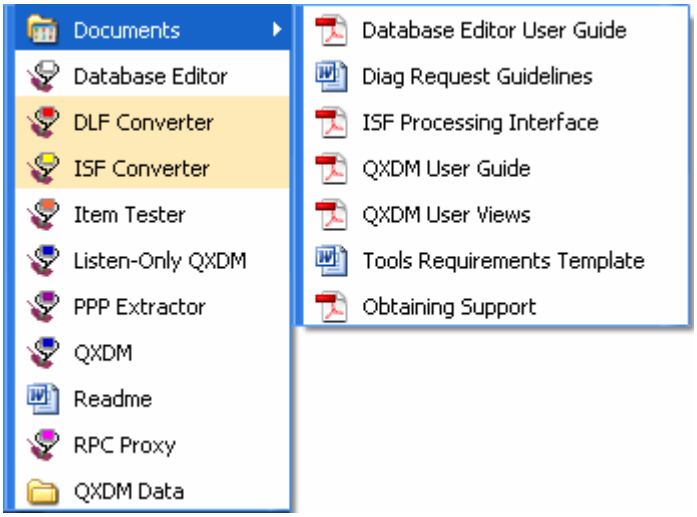

Figure 2-2 Start Menu – QXDM Tools Suite

2.4.2.1 Database Editor

The Database Editor provides an interface to describe user-defined items using QXDM user databases. Refer to [Q6] for complete documentation.

2.4.2.2 DLF Converter

The DLF Converter converts legacy DLF log files into ISF log files that can be loaded into QXDM for analysis. Command line support is also provided using the following syntax:

```
DLFConverter <DLF Input Filename> <ISF Output Filename>
```

2.4.2.3 ISF Converter

The ISF Converter converts ISF log files into legacy DLF log files for third-party DLF parsing support tools. Command line support is also provided using the following syntax:

```
ISFConverter <-pc> <ISF Input Filename> <DLF Output Filename>
```

The first option instructs the application to write out a PC timestamp in the generated log header for each item that does not contain a target timestamp.

2.4.2.4 Item Tester

The Item Tester is useful for viewing and testing items that are described in QXDM databases. It also provides legacy script command examples for all items described in the QXDM and user databases (see Section 7.2).

2.4.2.5 Listen-Only QXDM

Running Listen-Only QXDM starts QXDM in a Listen-Only mode. QXDM will not send configuration commands to the target when started this way.

2.4.2.6 QXDM

Selecting QXDM causes the main QXDM application to run.

2.4.2.7 PPP Extractor

PPP Extractor converts PPP logs generated by the phone to the format specified in RFC 1662. This tool is provided for legacy targets and operates by processing each of log code pairs in Table 2-2 to a file in the PPP dump format.

Table 2-2 PPP Extractor operation

Log Codes (RX, TX)	Description
0x109C, 0x109D	Legacy PPP logs
0x1113, 0x1123	UM interface legacy PPP logs
0x1114, 0x1124	RM interface legacy PPP logs
0x1115, 0x1125	AN interface legacy PPP logs

NOTE The tool will generate a file regardless of whether any instances of the above log code pairs exist in the ISF file being processed. In this case, the generated file or files will be empty and should not be loaded into a network protocol analyzer application.

NOTE For newer targets, the QXDM data protocol logging display and PCAP generator tool included with QCAT should be used instead of the PPP Extractor.

2.4.2.8 RPC proxy

This tool is used as a RPC-like proxy server for physical layer tests using an internal QUALCOMM proprietary interface.

2.4.3 QXDM documentation

The documents described in this section are installed with QXDM and can be accessed from the Windows Start menu. [Q6] and [Q8] are also available through the QXDM Help menu.

2.4.3.1 QUALCOMM ISF Processing Interface

The specification (see [Q7]) provides documentation for the postprocessing and analysis automation interface that supports both dynamic parsing DLLs and user database definitions (with preference towards the latter as it provides a richer interface). This interface does not differ in any substantial way from the real-time automation interface QXDM utilizes to implement roughly 60% of the existing QXDM displays.

2.4.3.2 QXDM Database Editor User Guide

This user guide (see [Q6]) describes how to use the Database Editor to provide packet item definitions for real-time QXDM parsing support.

2.4.3.3 ReadMe

Revision changes for each version of QXDM are detailed in the ReadMe file.

2.5 Item Store Format (.ISF) files

While running, QXDM generates a temporary log file called the Item Store, which ensures that data is not lost in the event of unexpected program termination. Under normal circumstances, this file is deleted upon termination. The Item Store is created in the QXDM ISF folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\ISF).

The Item Store contains all traffic that occurs between QXDM and the target (or targets if QXDM sequentially connects to more than one target in a given session) or since the last time the Item Store was cleared, e.g., via the Clear Items (CTRL + I) menu command. This provides an accurate representation of the state of the application at any given time for analysis and debugging.

2.5.1 Always-On logging

QXDM is always generating an ISF log file. Log files can be processed by QXDM at a later time for post-processing analysis and replay. Always-On logging behavior can be customized from the Item Store Settings command. For a complete description, refer to Section 3.1.10.

The Application Statistics view is the place to go for monitoring the status of logging (Section 4.5.1).

NOTE An ISF conversion utility is included with QXDM and is available from the Tools menu (Tools → ISF File Converter) for backwards compatibility with third-party tools that only support the legacy .DLF log file format.

NOTE Conversion between formats is a destructive process, i.e., information in the ISF will be lost (such as per-item PC timestamps and per-item parsing hints).

2.5.2 Recovering the Item Store

If, for some reason, QXDM is terminated abnormally, it is possible to recover the session. Section 3.1.6 describes how to load an ISF file. In the event of abnormal termination, it may be possible to recover and repair the lost session. The temporary ISF is located in the QXDM ISF folder.

2.5.3 Saving the Item Store

By default, the ISF is deleted upon QXDM termination. If, however, you wish to save a session for later analysis or viewing, check Enable Query For ISF Save (see Figure 3-9).

3 Display Overview

QXDM uses a multiple document interface to support any number of views simultaneously.

3.1 File menu

These options are available through the File menu selection.

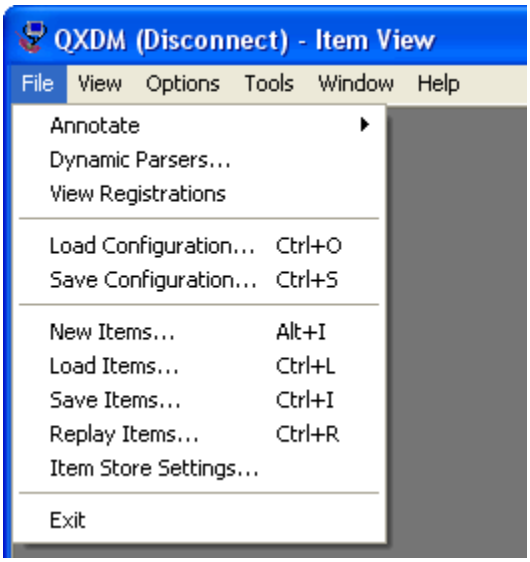

Figure 3-1 File menu options

3.1.1 Annotate

Predefined strings can be appended to the ISF using this command (see Section 2.5). Users can override default annotation strings by including a UserAnnotation.txt table in the Microsoft Windows shared documents folder for all users typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\Database. To replace existing annotations described in Annotation.txt, copy one or more lines using a text editor like Notepad and save the changed text in a file named UserAnnotation.txt.

NOTE Do not edit the Annotation.txt file directly as it is overwritten each time QXDM is installed.

3.1.2 Dynamic Parsers

QXDM supports the addition of dynamic parsers that the end user can provide to parse items. Refer to Chapter 9 for information on creating runtime parsers. Use the Parsing Engines window to tell QXDM where to look to load these parsers (see Figure 3-2).

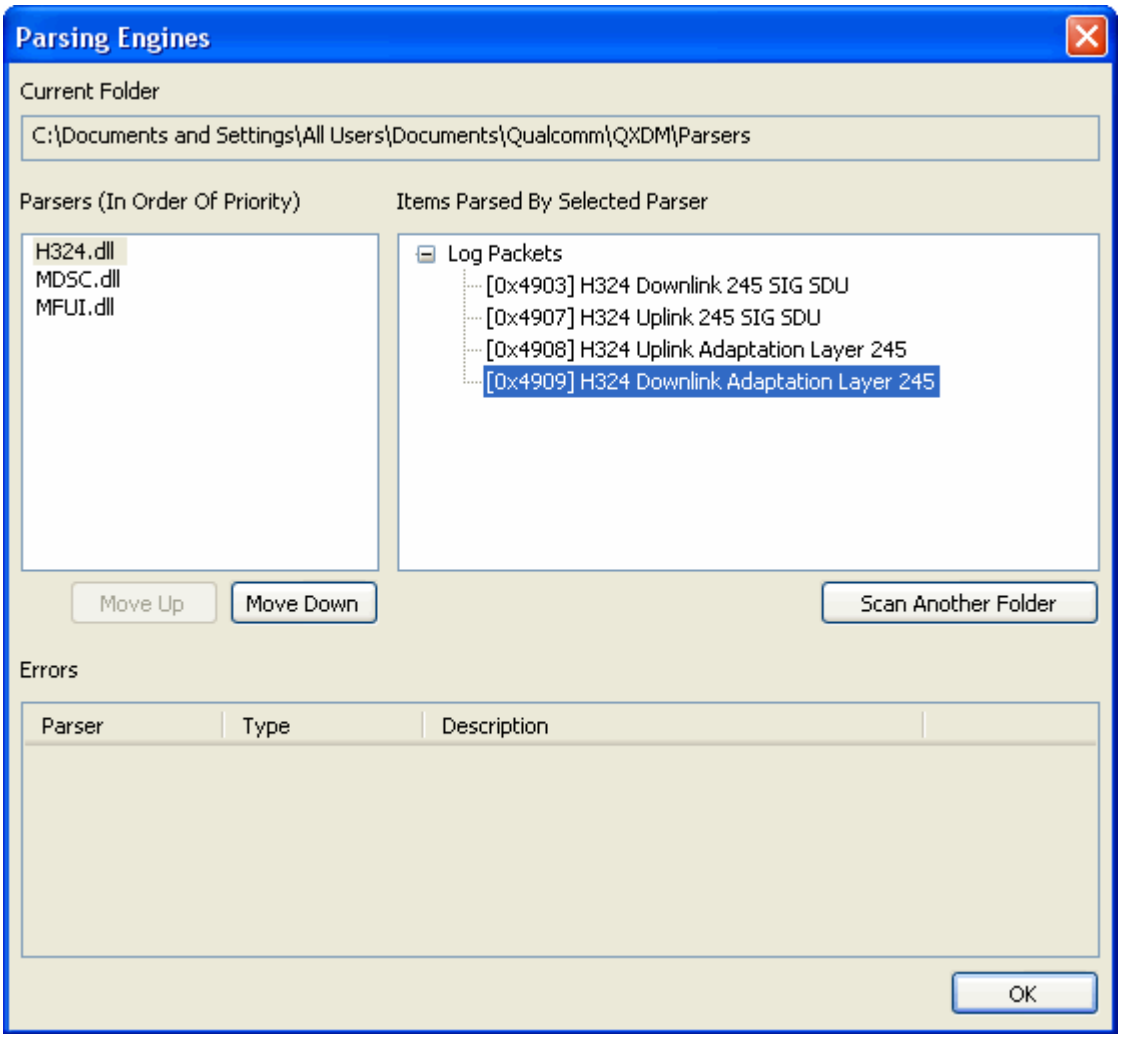

Figure 3-2 File → Dynamic Parsers…

3.1.3 View Registrations

Views register interest in items when they are opened and unregister when they are closed, e.g., opening the Finger Placement view results in registration for the Searcher Finger and Enhanced Finger Information logs. Figure 3-3 demonstrates using View Registrations to see what items have been registered by the current set of open Views.

This view is also useful in determining if an item has been registered with the phone, e.g., if you have requested a log that the phone is not sending, check View Registrations. If the log is checked, then contact the responsible phone technology or build person for help in determining why the phone is not sending the log.

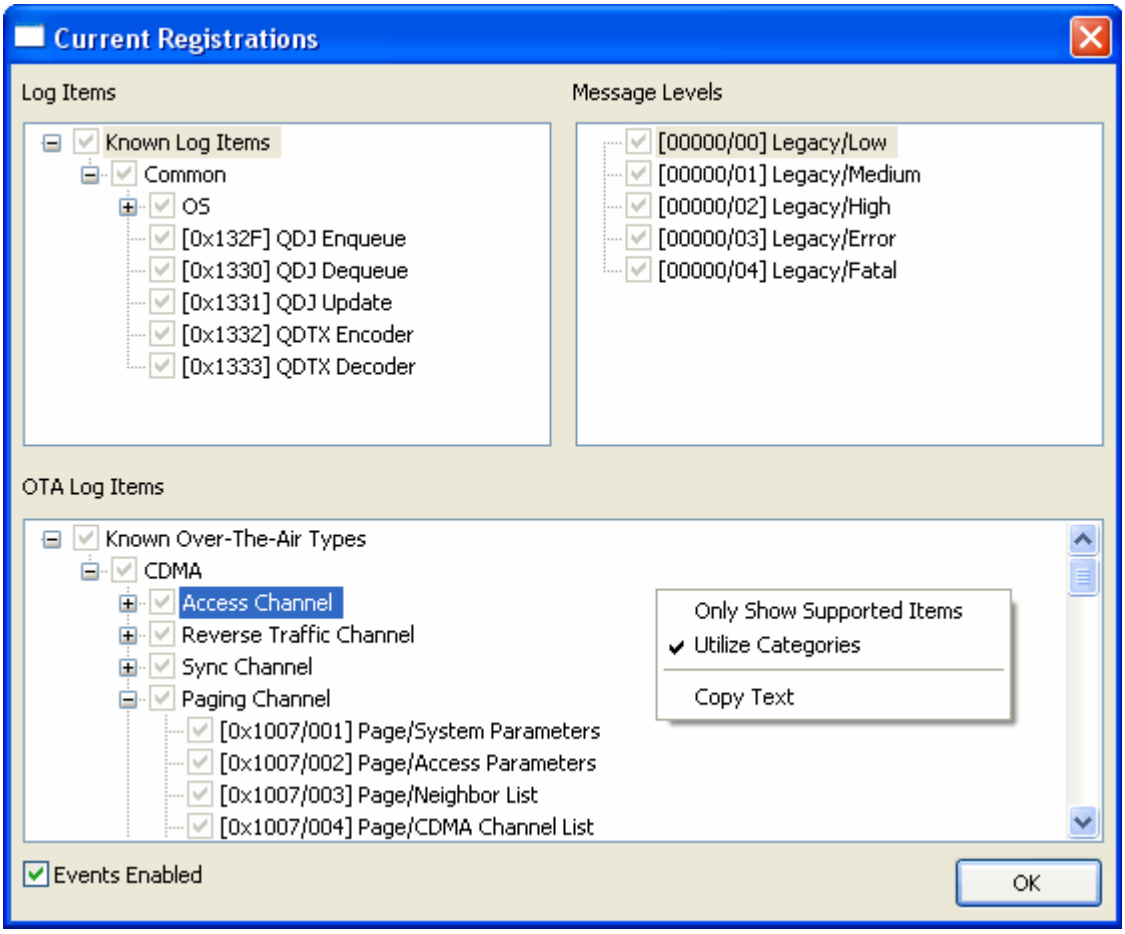

Figure 3-3 File → View Registrations…

3.1.3.1 Context options

Additional context-sensitive options allow you to refine the controls and can be accessed by clicking the right mouse button over the view area to select the following items:

- Only Show Supported Items – When checked, only items supported by the phone are displayed
- Utilize Categories – When unchecked, all items are shown in a single list; this can sometimes be helpful locating an item
- Copy Text – When selected, the registrations for the clicked item type are converted to text and copied to the clipboard.

3.1.4 Load Configuration…

Previously saved configurations can be loaded using the Load Configuration command. Configurations such as selected views and registrations are restored from QXDM configuration files (.DMC extension).

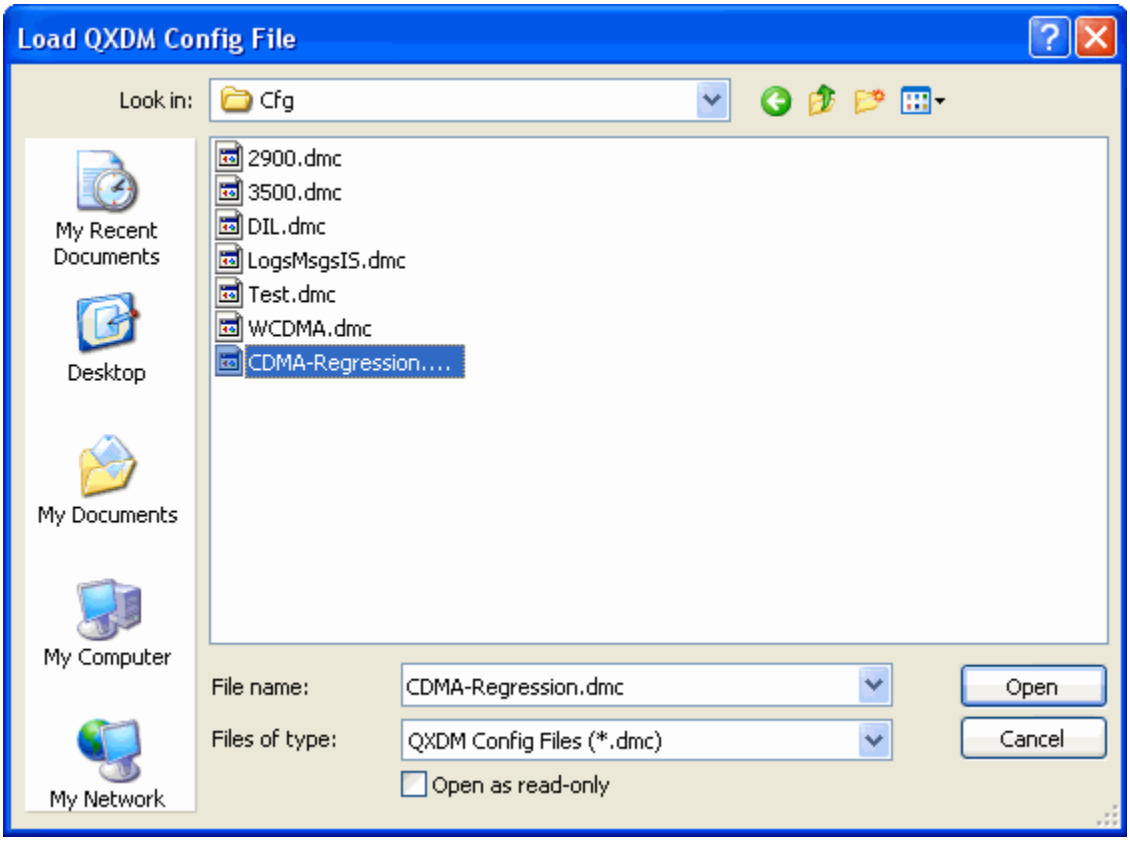

Figure 3-4 File → Load Configuration…

3.1.5 Save Configuration…

Configurations remember selected view registrations and settings and can be saved to QXDM configuration files (.DMC extension) for later loading.

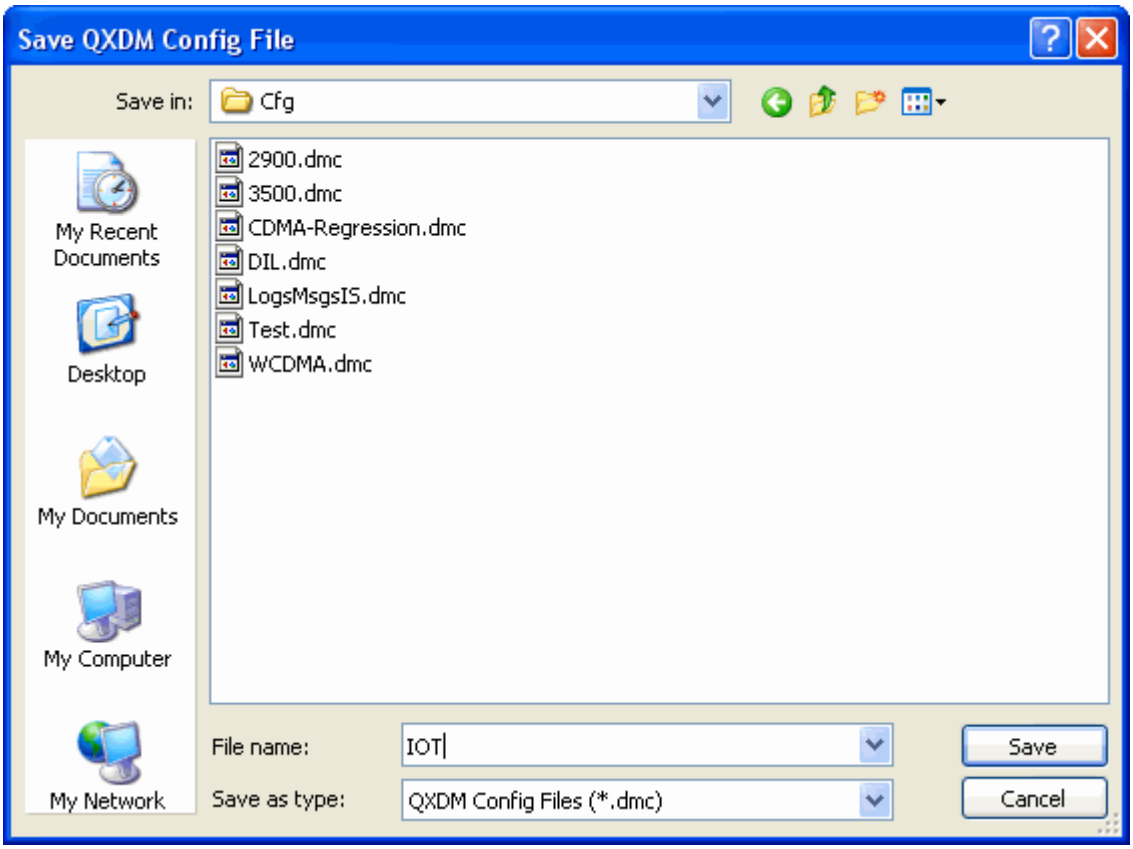

Figure 3-5 File → Save Configuration…

3.1.6 New Items…

New Items creates a new temporary ISF. This is useful if you have been reviewing an existing file store (by loading or replaying it) and now want to connect to a live target without appending to the current file. This option can also be used to start a new ISF in advance of commencing a test.

NOTE If the Enable Query For ISF Save option (Figure 3-9) has been set, QXDM will prompt you to save the current temporary ISF before creating a new ISF. If this option has not been set, the current temporary ISF will be automatically deleted.

3.1.7 Load Items…

NOTE This command will be enabled only if QXDM is not connected to a target.

A previously saved .ISF can be loaded for viewing.

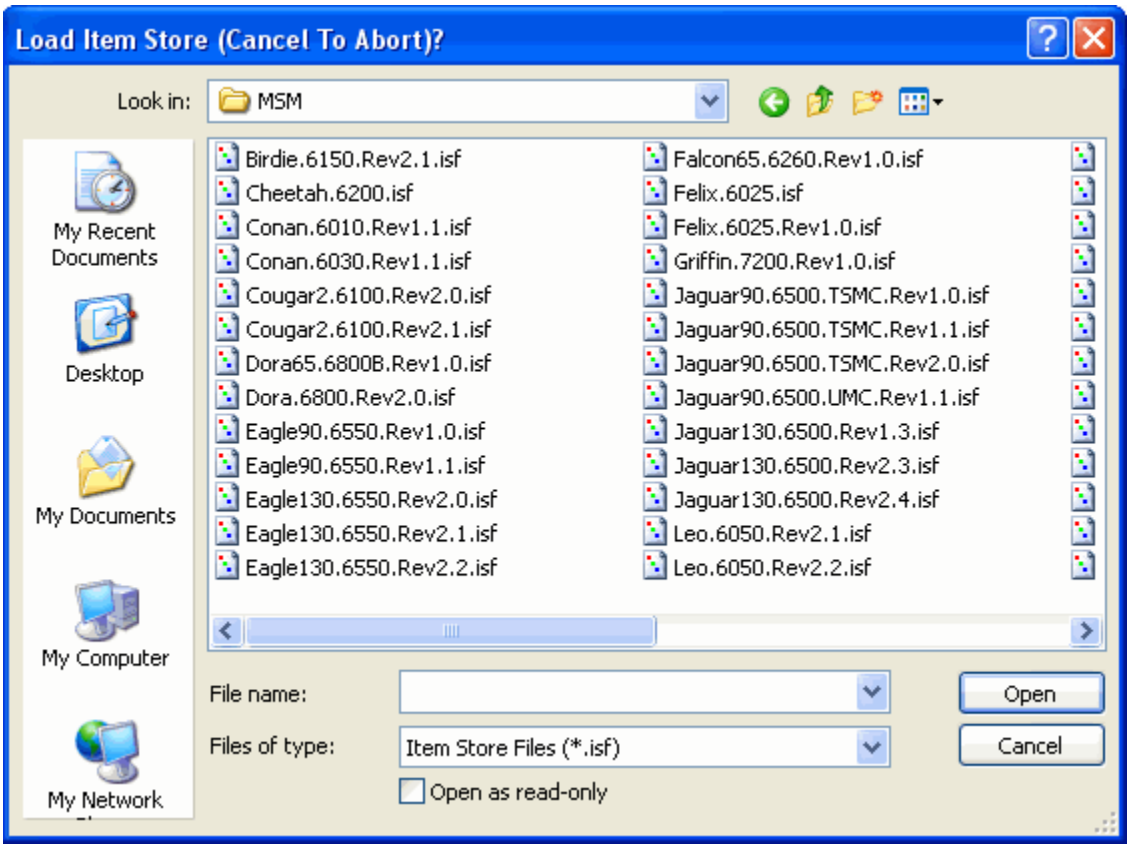

Figure 3-6 File → Load Items…

3.1.8 Save Items…

Save Items behaves exactly the same as New Items except that there is no check to see if Enable Query For ISF Save (Figure 3-9) is checked.

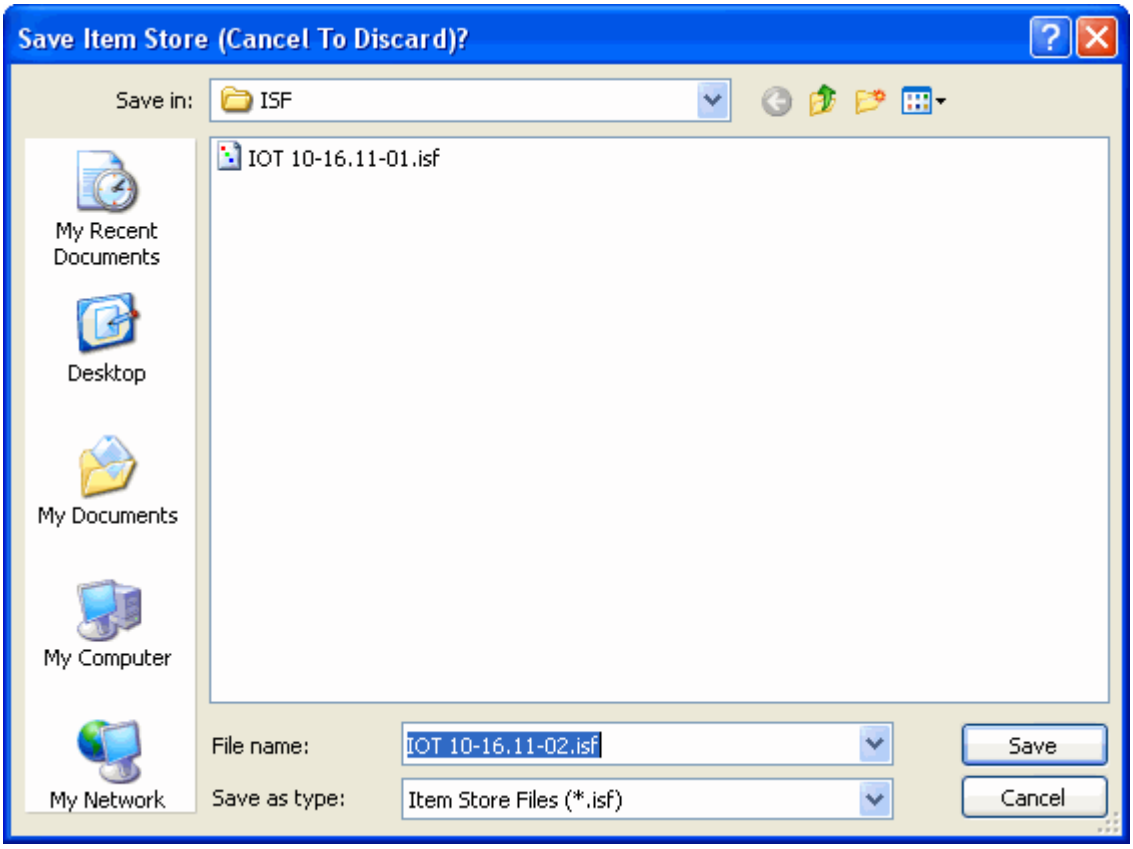

Figure 3-7 File → Save Items…

3.1.9 Replay Items…

NOTE This command will be enabled only if QXDM is not connected to a target.

A previously saved ISF can be loaded for replaying using this command.

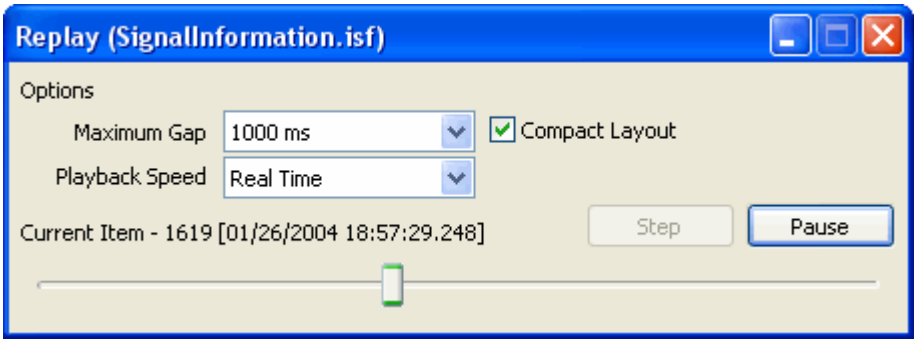

Figure 3-8 Replay Items

3.1.9.1 Maximum Gap

Set the maximum amount of time in milliseconds to wait before playing the next item. If the actual gap exceeds the configured value, the configured value will be used as the amount of time to wait until playing the next item.

3.1.9.2 Playback Speed

Set the speed at which items are played.

NOTE Using the No Wait option can impact system performance when one has views open that process each and every item.

3.1.9.3 Compact Layout

Information about an item is displayed as it is processed unless Compact Layout is checked.

3.1.9.4 Step

Replay just one item when STEP is pressed.

3.1.9.5 Play/Pause

Replay based on selected settings when PLAY is pressed. Stop replaying when PAUSE is pressed.

3.1.10 Item Store Settings…

The Item Store Settings dialog allows the user to control ISF logging behavior.

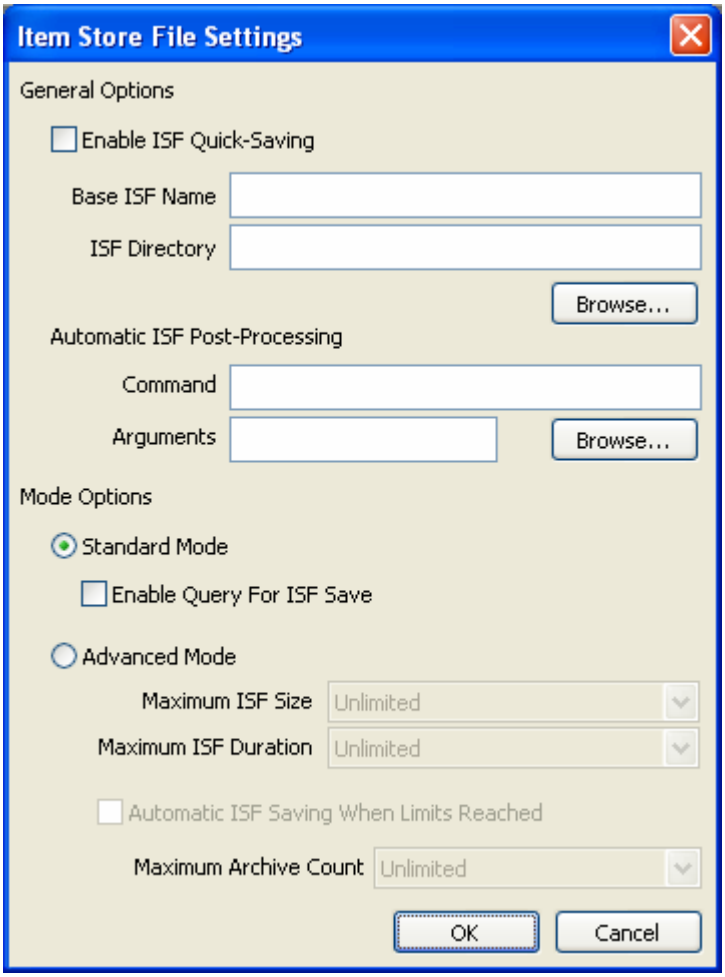

Figure 3-9 Item Store File Settings

3.1.10.1 General Options

The default behavior is for QXDM to prompt for a log file name on user-initiated log file saves (e.g. File → Save → Items, Section 3.1.8).

3.1.10.1.1 Enable ISF Quick-Saving

When Enable ISF Quick-Saving is checked, the log file is saved using the Base ISF Name at the ISF Directory location, without prompting the user for a filename.

NOTE This applies only to scenarios where the user has initiated the save, i.e., selected the Save Items menu command. In the other scenarios, where an ISF needs to be saved due to a parallel operation (New Items, Load Items, and exiting QXDM), the user may still be prompted for a filename (dependent on the mode). In all scenarios, an automatically generated name will appear in the prompt dialog itself.

3.1.10.1.1.1 Base ISF Name

Use Base ISF Name to describe the unique portion of the ISF filename that will be used when in Advanced Mode. Using the base ISF name, QXDM will format the name as follows:

```
<Base Name>MM-DD-HH-MM.isf
```

3.1.10.1.1.2 ISF Directory

Use ISF Directory to describe where to save ISF files. The default is located in the Microsoft Windows shared documents folder for all users typically at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\ISF.

3.1.10.1.2 Automatic ISF Post-Processing

Automatic post-processing of log files is supported by providing a command and optional arguments that QXDM will run each time a log is saved. The log is passed to the specified command by QXDM.

3.1.10.1.2.1 Command

Enter the command including the fully qualified path that QXDM is to run, e.g., C:\WINDOWS\System32\CScript.exe.

3.1.10.1.2.2 Arguments

Enter any required arguments for the command, e.g., C:\TestScripts\ISFAnalyzeLog1234.js. The format requirements are the same as if run from a command window, i.e., if the argument contains spaces, it should be enclosed in quotes.

3.1.10.2 Mode Options

The default logging mode is for QXDM to delete the temporary log file upon program termination.

3.1.10.2.1 Standard Mode

Selecting Standard Mode results in the default behavior described above.

3.1.10.2.1.1 Enable Query For ISF Save

When checked and in Standard Mode, the user is prompted to save the temporary Item Store upon creating a new Item Store (New Items), loading an Item Store (Load Items), or before exiting.

3.1.10.2.2 Advanced Mode

Selecting this option causes QXDM to apply the advanced options described below.

3.1.10.2.2.1 Maximum ISF Size

Use Maximum ISF Size to set the maximum file size the current temporary ISF can reach before it is closed and a new temporary ISF is used.

3.1.10.2.2.2 Maximum ISF Duration

Use Maximum ISF Duration to control the maximum file duration (time between first item's generic timestamp and last item's generic timestamp) that can elapse before the current temporary ISF is closed and a new ISF is used.

3.1.10.2.2.3 Automatic ISF Saving When Limits Reached

This checkbox controls whether an ISF that has exceeded specified limits is automatically saved. If unchecked, the temporary ISF is deleted when limits are reached.

3.1.10.2.2.4 Maximum Archive Count

Use the Maximum Archive Count to identify the maximum number of ISF files that will be saved. When this value is exceeded, the oldest ISF saved by the current QXDM process is deleted.

3.1.11 Exit

Click EXIT to end the QXDM session.

3.2 View menu

These options are available through the View menu selection.

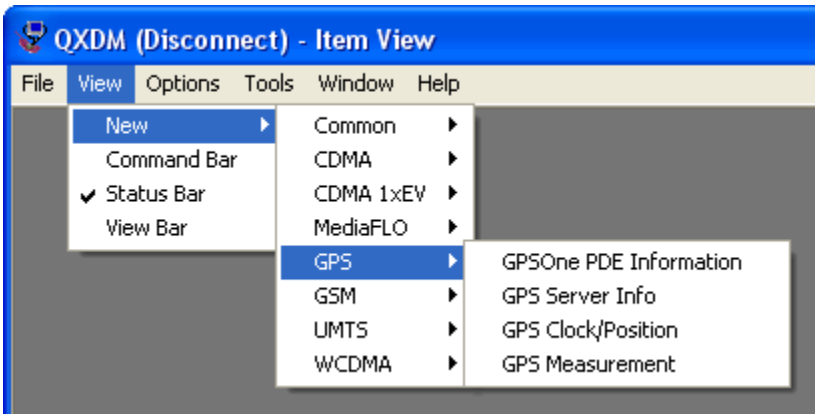

Figure 3-10 View menu

3.2.1 New

Use this option to select a view grouped by logical categories.

3.2.2 Command Bar

Select this option to display or hide the Command Bar.

3.2.3 Status Bar

Select this option to display or hide the Status Bar.

3.2.4 View Bar

Select this option to display or hide the View Bar.

3.3 Options menu

These options are available through the Options menu selection.

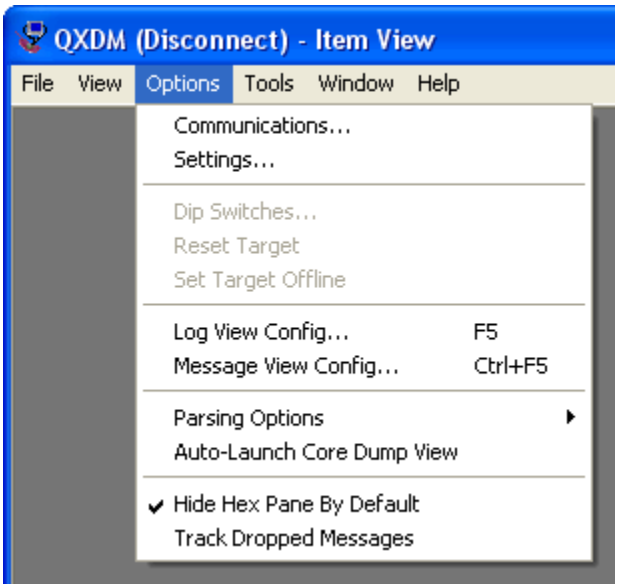

Figure 3-11 Options menu

3.3.1 Communications…

The Communications dialog, shown in Figure 3-12, allows you to configure the COM ports that will be used by QXDM for connection purposes.

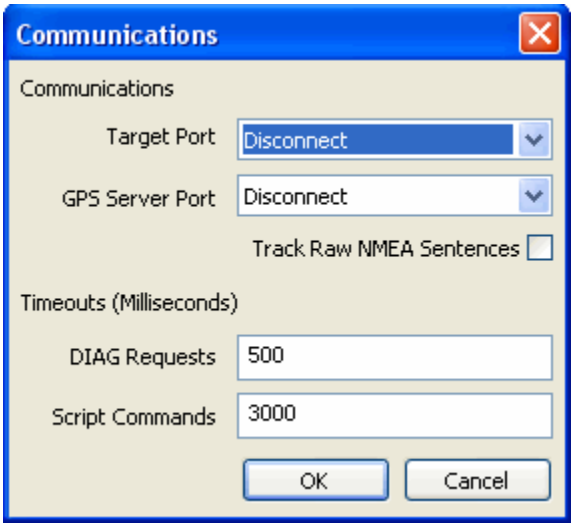

Figure 3-12 Options → Communications… dialog

3.3.1.1 Target Port

Target Port lists the union of COM ports supported by the QPST Port Server and the PC hardware. Double-click the blue globe at the bottom right side of the Windows Task Bar, illustrated in Figure 3-13, to configure the ports that the QPST Port Server is monitoring.

Figure 3-13 QPST configuration globe

The QPST configuration utility is shown in Figure 3-14.

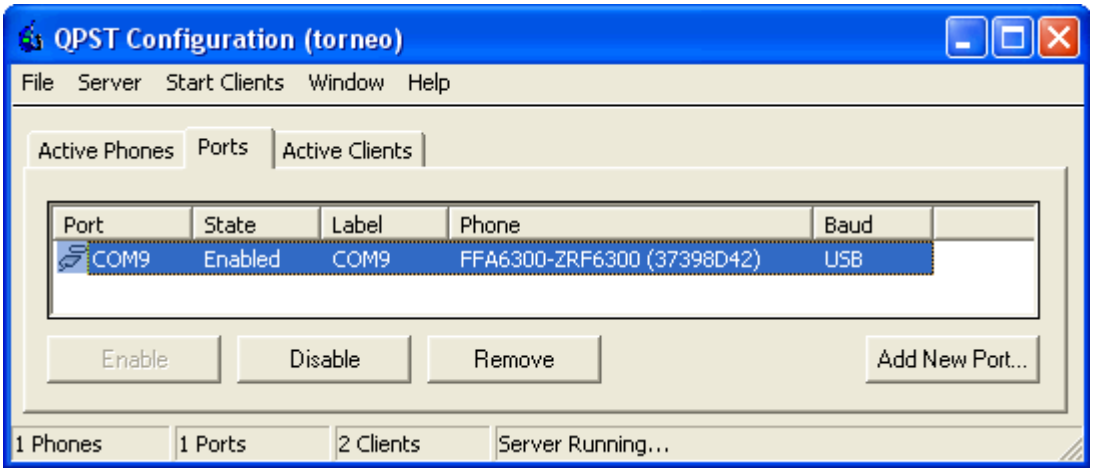

Figure 3-14 QPST Port Server configuration

3.3.1.2 GPS Server Port

Select the COM port on which your GPS receiver is connected. QXDM will start collecting GPS information from the GPS receiver when connection is established.

NOTE Be sure to remove the COM port that your GPS receiver is connected to from the list of ports that QPST Configuration is monitoring. Otherwise, the COM port will not be visible to QXDM.

QXDM expects COM ports connected to GPS receivers to be configured to the appropriate values (typically 4800, 8, N, 1) via the Windows Device Manager.

Select the Track Raw NMEA Sentences option to add raw NMEA sentences to the current ISF log file in addition to the finalized GPS information obtained by processing valid/recognized NMEA sentences.

3.3.1.3 Timeouts (MS)

DIAG requests

This is the time, in ms, that QXDM waits for a reply from the phone to DIAG requests.

NOTE The maximum value QXDM allows is 2000. The actual timeout is three times the value specified. The QPST Port Server automatically retries requests up to three times, so a timeout of 2 sec actually translates to a net timeout of 6 sec.

Script commands

This is the time, in milliseconds, that QXDM waits before timing out for legacy script commands.

3.3.2 Settings…

This dialog shown in Figure 3-15 allows you to set various personal preference settings.

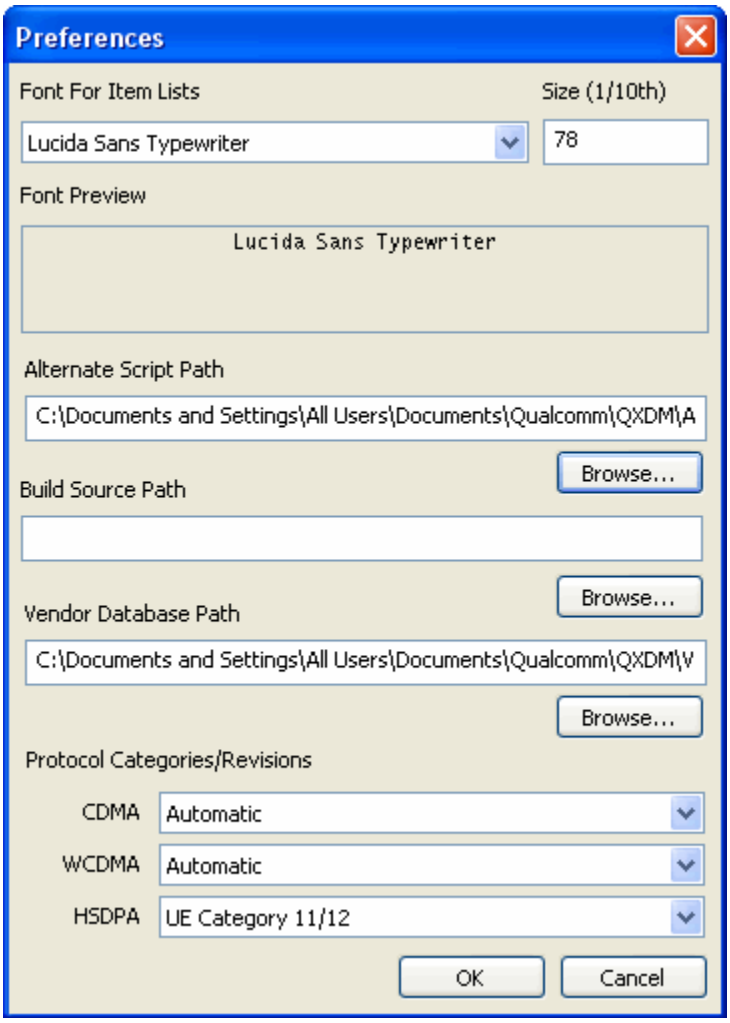

Figure 3-15 Options → Settings… dialog

3.3.2.1 Font Settings

Using these settings, you can set the font face and size, specified in tenths of a point, that is used in all QXDM displays or specify the display components that use a fixed width font by default. This includes the following displays, display components, and dialogs:

- Memory Viewer display
- Task list pane of Task Profiling – APU/MPU displays
- Dynamic Item List display
- IS-95A Retrievable Parameters display
- Each pane (scrolling list, raw item, and parsed item pane) of all item list-based displays (Command Output, Messages View, Log View, Item View, and Filtered View)
- Raw Item Contents dialog

A preview of the selected font is displayed in the preview pane.

NOTE The allowable range for the font face size is 80 – 240.

NOTE If a font is not specified, the default font face used is Lucida Console and the default font size is taken from the current Microsoft Windows® icon font size.

NOTE It is not strictly necessary to choose a fixed width font. For displays or display components that are designed around a fixed width font (such as the Memory Viewer display), the font face selection will be ignored when a variable width font is chosen. Instead, the default font face as given above will be used.

3.3.2.2 Alternate Script Path

This provides an alternate path from which to run scripts; the default is the QXDM application folder (typically C:\Program Files\Qualcomm\QXDM\bin).

3.3.2.3 Build Source Path

Use Build Source Path to supply a path that QXDM will search when a request is made to view the source file associated with a debug message (see Section 4.2.1.5.4).

3.3.2.4 Vendor Database Path

Use Vendor Database Path to supply a path that QXDM will use to locate a vendor's database path. A vendor-specific database is used when parsing items specific to a particular handset vendor (currently only NV items). A vendor-specific database can also contain mobile model names.

3.3.2.5 WCDMA Protocol Revision

This is used to control the protocol revision utilized by QXDM when parsing WCDMA Over-the-Air (OTA) messages. Item Store files created by QXDM version 03.09.15 or later utilize the WCDMA protocol revision reported by the target. For these files (and real-time interaction), the Automatic choice is the best selection. For files created by QXDM versions prior to 03.09.15, the option should be set to the appropriate protocol revision.

NOTE Utilizing the Automatic choice when viewing an ISF created by QXDM versions prior to 03.09.15 is equivalent to choosing 06/04 V5.9.0.

3.3.2.6 CDMA protocol revision

This is used to control the protocol revision utilized by QXDM when parsing CDMA OTA messages. Item Store files created by QXDM version 03.08.99 or later utilize the most recent CDMA protocol revision reported by the target. For these files (and real-time interaction), the Automatic choice is the best selection. For files created by QXDM versions prior to 03.08.99 the option should be set to the appropriate protocol revision.

NOTE Utilizing the Automatic choice when viewing an ISF created by QXDM versions prior to 03.08.99 is equivalent to choosing [06] – IS-2000 Rev 0.

3.3.2.7 HSDPA UE Category

This setting allows you to control the HSDPA UE category used when the QXDM HSDPA displays need to map CQI indices to bit sizes. This value should match the UE category of the connected target.

3.3.3 DIP Switches…

This option allows you to configure software DIP switch settings using the diagnostic services Get/Set DIP switch commands, as shown in Figure 3-16. Not all targets support this display.

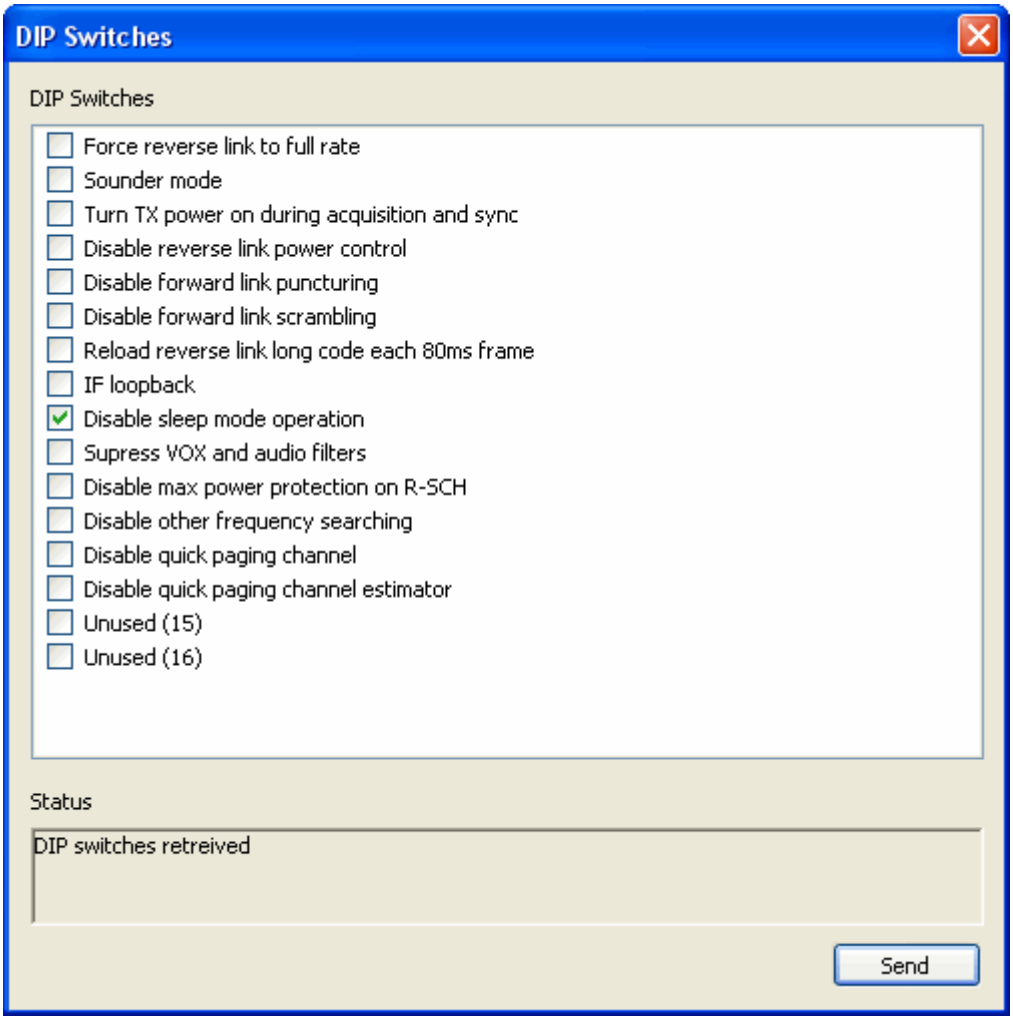

Figure 3-16 Options → DIP Switches… dialog

3.3.4 Log View Config…

NOTE The legacy Log View has limited functionality and is provided only for backward compatibility. The correct approach is to use Always-On logging behavior in conjunction with Filtered Views.

Refer to Section 3.1.10 for details on how to configure Always-On logging behavior.

3.3.5 Message View Config…

Although Message View Config is still supported, Filtered Views eliminate the need for a dedicated Message View Config dialog.

NOTE The legacy Messages View has limited functionality and is provided only for backwards compatibility. The correct approach is to use Filtered Views and configuration (.DMC) files.

3.3.6 Parsing Options

Parsing Options allows you to control how QXDM parses items. Using these options shown in Figure 3-17, you can control exporting parsed text and the order that QXDM uses to select available parsing solutions.

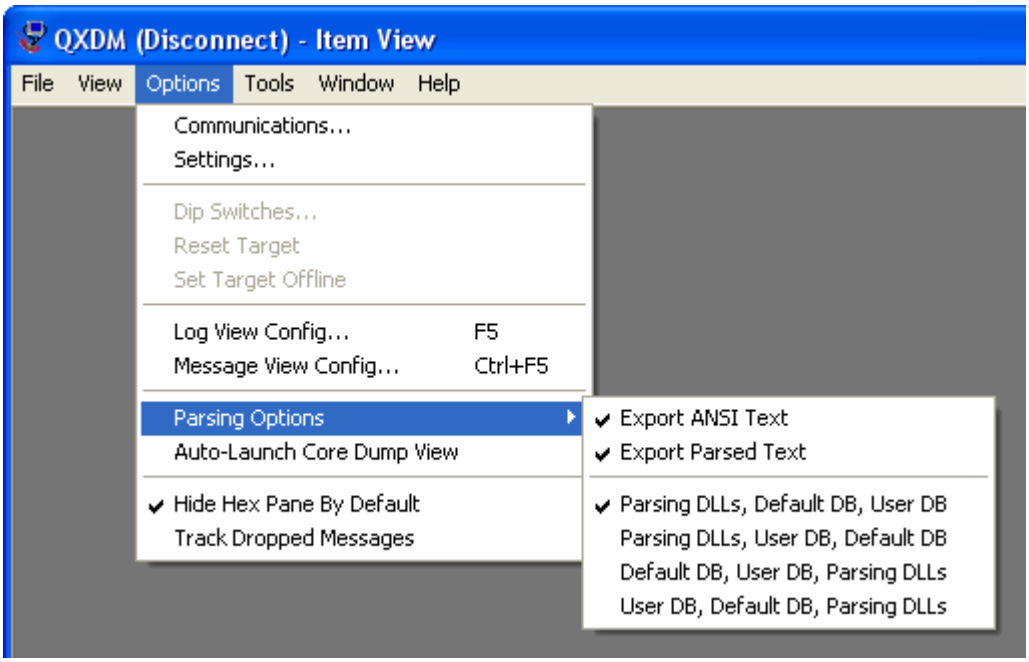

Figure 3-17 Parsing Options

3.3.6.1 Export ANSI Text

Force text exports to ANSI (as opposed to Unicode).

3.3.6.2 Export Parsed Text

Selecting this menu item enables the exporting of parsed text to a file or to the clipboard. Parsed text is any additional output generated by dynamic parsers like SILK and user-provided DLLs (see Chapter 9).

3.3.6.3 Parsing DLLs, Default DB, User DB

Selecting this menu item causes QXDM to parse items in the following order preference: Parsing DLLs, QXDM database, User databases.

3.3.6.4 Parsing DLLs, User DB, Default DB

Selecting this menu item causes QXDM to parse items in the following order preference: Parsing DLLs, User databases, QXDM database.

3.3.6.5 Default DB, User DB, Parsing DLLs

Selecting this menu item causes QXDM to parse items in the following order preference: QXDM database, User databases, Parsing DLLs.

3.3.6.6 User DB, Default DB, Parsing DLLs

Selecting this menu item causes QXDM to parse items in the following order preference: User databases, QXDM database, Parsing DLLs.

3.3.7 Auto-Launch Core Dump View

Deselecting this menu item disables the automatic launching of the OS Core Dump View. By default, this is enabled so that on receipt of an OS Core dump log from supporting targets, the contents are displayed.

3.3.8 Hide Hex Pane By Default

Checking this menu item causes the hexadecimal bytes display portion (lower left-hand pane) of filtered views to be hidden initially.

3.3.9 Track Dropped Messages

Checking this menu item causes QXDM to include dropped informational messages reported by the phone from each debug message response packet.

3.4 Tools menu

This is available through the Tools menu selection and provides a link to other installed tools.

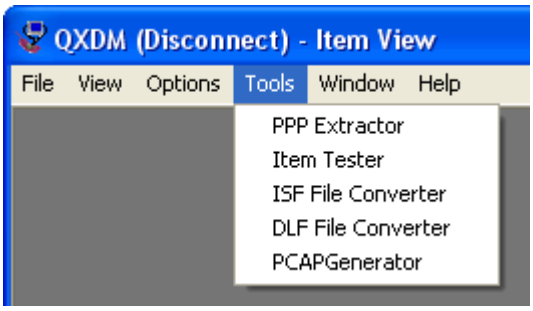

Figure 3-18 Tools menu

This is available through the Window menu selection and allows you to manage active windows.

3.5 Window menu

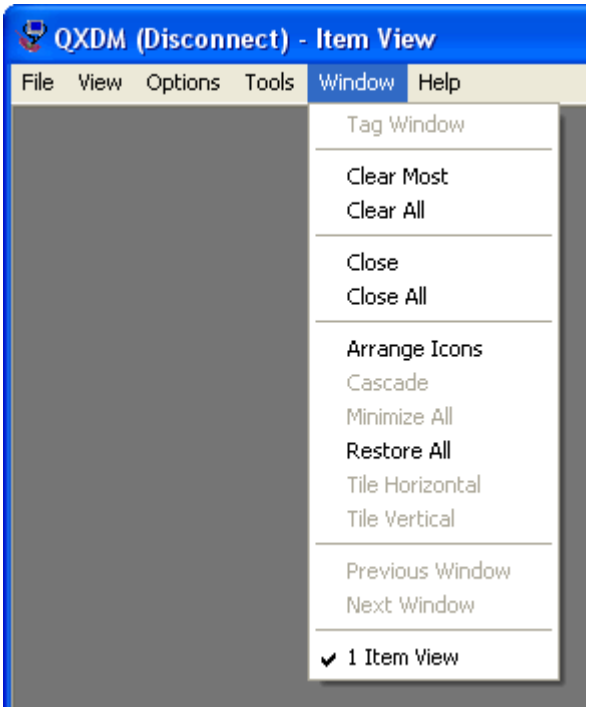

Figure 3-19 Window menu

3.5.1 Tag Window

By selecting this option, Filtered views can be tagged with user-customizable names.

3.5.2 Clear Most

Selecting this option will result in all views that support the clear command being cleared, excluding the Item View.

NOTE The legacy Log View is unaffected by this command.

3.5.3 Clear All

Selecting this option will result in all views that support the clear command being cleared.

NOTE Using this command will result in all data collected in a QXDM session being lost.

3.5.4 Close

When this option is selected, the window in focus will be closed.

3.5.5 Close All

When this option is selected, all windows will be closed.

3.5.6 Arrange Icons

When this option is selected, windows that have been minimized will be rearranged.

3.5.7 Cascade

When this option is selected, windows that are opened will be rearranged in a cascading order.

3.5.8 Minimize All

When this option is selected, all open windows will be minimized.

3.5.9 Restore All

When this option is selected, all minimized windows will be restored to their open settings.

3.5.10 Tile Horizontal/Tile Vertical

When this option is selected, windows that are opened will be rearranged in a vertical or horizontal fashion.

3.5.11 Previous Window/Next Window

When this option is selected, focus will be set to the next or previous logical window.

3.5.12 Active Window List

Active Window List displays all active windows. Clicking a window in this list will cause the focus to change to the selected window.

3.6 Help menu

The Help menu provides access to licensing commands as well as links to the QXDM User Guide, the QXDM Release Notes and the support information.

Figure 3-20 Help menu

3.6.1 Licensing

Licensing allows you to view the status of the QUALCOMM License Management System (QLMS) license. If you need to temporarily release the QXDM QLMS license on the current computer, select Deactivate.

NOTE QLMS license deactivation requires an Internet connection. If you attempt to deactivate the QLMS license without an Internet connection, the license will be removed from the current computer but will still remain in the QLMS system. In this case, the only way to reclaim the license for use on another computer is to contact QLMS support and have the license manually removed from the QLMS system.

3.7 View Bar

The View Bar provides access to all views supported by QXDM.

Figure 3-21 View Bar

3.8 Command Bar

The Command Bar provides access to the legacy QXDM command interface.

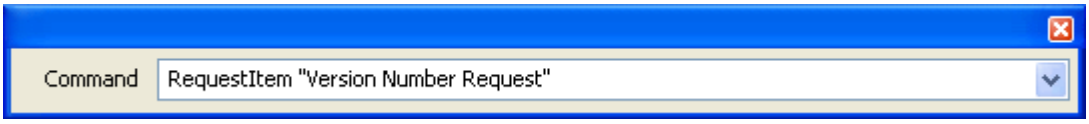

Figure 3-22 Command Bar

3.8.1 Command interface

Although many legacy DOS DM script commands are supported through the QXDM command interface, the recommended approach for automated testing is to use the automation interface (see Chapter 8).

Script commands can be typed in to the Command edit control. The Command Output display and the Item View display script output. Additionally, any Filtered View configured for the item type “Strings” and the item key “Automation” displays script output.

3.8.1.1 Using filenames

Some commands take filenames as arguments such as the Run command used to launch a script. To locate a file, QXDM will use the following search criteria:

1. Search using the file as provided
2. Search using the installation default QXDM data path (C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM)
3. Search using the Alternate Script path described in Options → Settings (Figure 3-15)

3.8.2 Script Help for command interface

The Script Help page documents all supported commands. It is available in the View combo-box from the View Bar (Figure 4-1). This view also provides documentation for the legacy script commands that are part of the original DOS DM scripting interface.

3.9 Status Bar

The Status Bar displays a configurable subset of application statistics, application state indicators, and help for select menu commands, as shown in Figure 3-23.

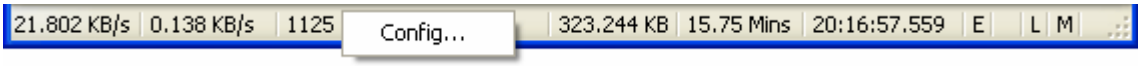

Figure 3-23 Status Bar

3.9.1 Config…

Right-clicking the Status Bar brings up an option allowing you to configure which indicators the Status Bar displays as well as the order of displayed indicators.

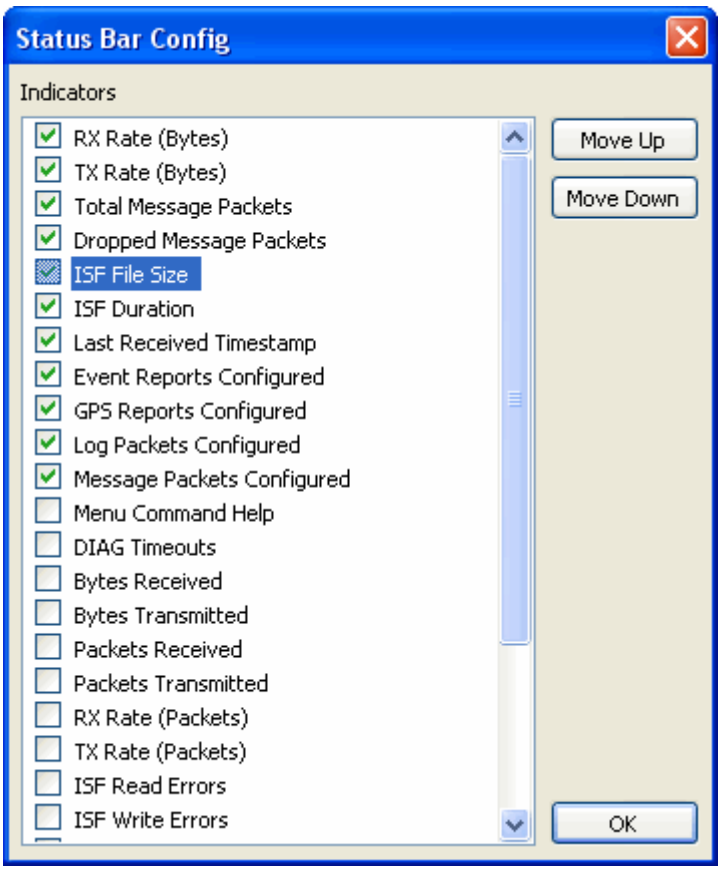

Figure 3-24 Status Bar Config

The status bar supports the indicators described in Table 3-1.

Table 3-1 Status Bar Indicators

Indicator	Description
Menu Command Help	Help for the currently selected menu item (when available)
DIAG Timeouts	Total number of response timeouts when sending DIAG requests
Bytes Received	Total number of bytes received over COM port
Bytes Transmitted	Total number of bytes transmitted over COM port
Packets Received	Total number of packets received over COM port
Packets Transmitted	Total number of packets transmitted over COM port
RX Rate (Bytes)	Rate at which data is being received over COM port
TX Rate (Bytes)	Rate at which data is being transmitted over COM port
RX Rate (Packets)	Rate at which packets are being received over COM port
TX Rate (Packets)	Rate at which packets are being transmitted over COM port
ISF File Size	Size of current ISF log file
ISF Duration	Amount of elapsed time current ISF represents
ISF Read Errors	Total number of errors reading an item from current ISF log file
ISF Write Errors	Total number of errors writing an item to current ISF log file
Last Generic Timestamp	Last generic (i.e., PC) timestamp generated
Last Received Timestamp	Last target timestamp received
Total Event Reports	Total number of event reports added to current ISF log file
Total GPS Reports	Total number of GPS reports added to current ISF log file
Total Log Packets	Total number of log packets added to current ISF log file
Total Message Packets	Total number of message packets added to current ISF log file
Dropped Message Packets	Total number of dropped message packets reported by target
Event Reports Configured	Is the target event service configured to generate traffic?
GPS Reports Configured	Is the QXDM GPS service configured?
Log Packets Configured	Is the target log service configured to generate traffic?
Message Packets Configured	Is the target debug message service configured to generate traffic?
GPS Latitude	Current GPS latitude (as reported by GPS receiver)
GPS Longitude	Current GPS longitude (as reported by GPS receiver)
GPS Time	Current GPS time (as reported by GPS receiver)

4 QXDM Views

4.1 QXDM views

QXDM views are accessed from the QXDM View bar, as shown in Figure 4-1, or the View menu.

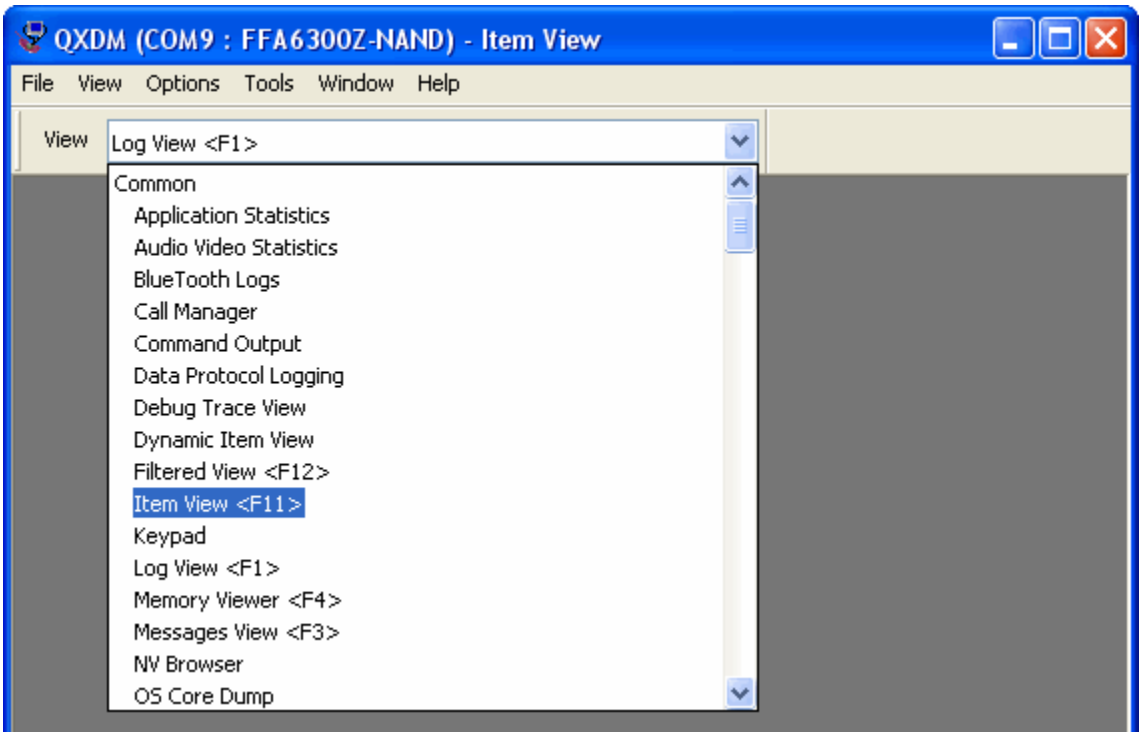

Figure 4-1 QXDM views using the View bar

4.2 Item list views

An item list view is a scrolling view that contains zero or more items that are derived from the contents of the current ISF log file. This includes the Item View, Filtered Views, the Messages View, the Log View, and the Command Output display.

The Item View is a special item list view that shows all items generated during a QXDM session. When QXDM is run, a temporary ISF is created in the QXDM ISF folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\ISF). The Item View shows the contents of this temporary ISF log file and can be displayed from the View Bar or by using the F11 accelerator key.

A Filtered View represents a subset of the contents of the current ISF and therefore the Item View. This subset is configured by item type and/or item key. Unlimited filtered views may be created by using the accelerator key F12 or selecting Filtered View from the View Bar. Filtered views can also be created by interacting with an existing item list based display through the Refilter Items, Match Items, and Process Items menu commands.

For legacy reasons, the Messages View and Log View are preserved as special case Filtered Views. The Messages View is a Filtered View that can be configured from the main QXDM menu without being active. The Log View is similar to the Messages View in that it can be configured without being active. In addition to this behavior the Log View implements additional functionality specific to legacy logging.

The Command Output display is a special case Filtered View whose configuration is fixed to only accept items whose type is String and whose key is Automation. Thus, the Command Output display shows all output from the legacy QXDM command interface and the preferred QXDM automation interface.

Excluding the legacy Messages View, Log View, and Command Output display, an item list view has three adjustable panes. The top pane is a scrolling list where a summary of each item in the view is displayed. The bottom left pane is used to display (in a hexadecimal dump) the raw data of the item currently selected in the scrolling list. The bottom right pane is used to display the parsed fields or (when available) the parsed text of the currently selected item in the scrolling list.

4.2.1 Scrolling List Pane

All item list based QXDM displays contain the Scrolling List Pane. This includes the Item View, Filtered Views, the Messages View, the Log View, and the Command Output display. This pane displays each item in the view, one per line in the list.

4.2.1.1 Menu options

Right-clicking within the Scrolling List Pane brings up the context-sensitive menu.

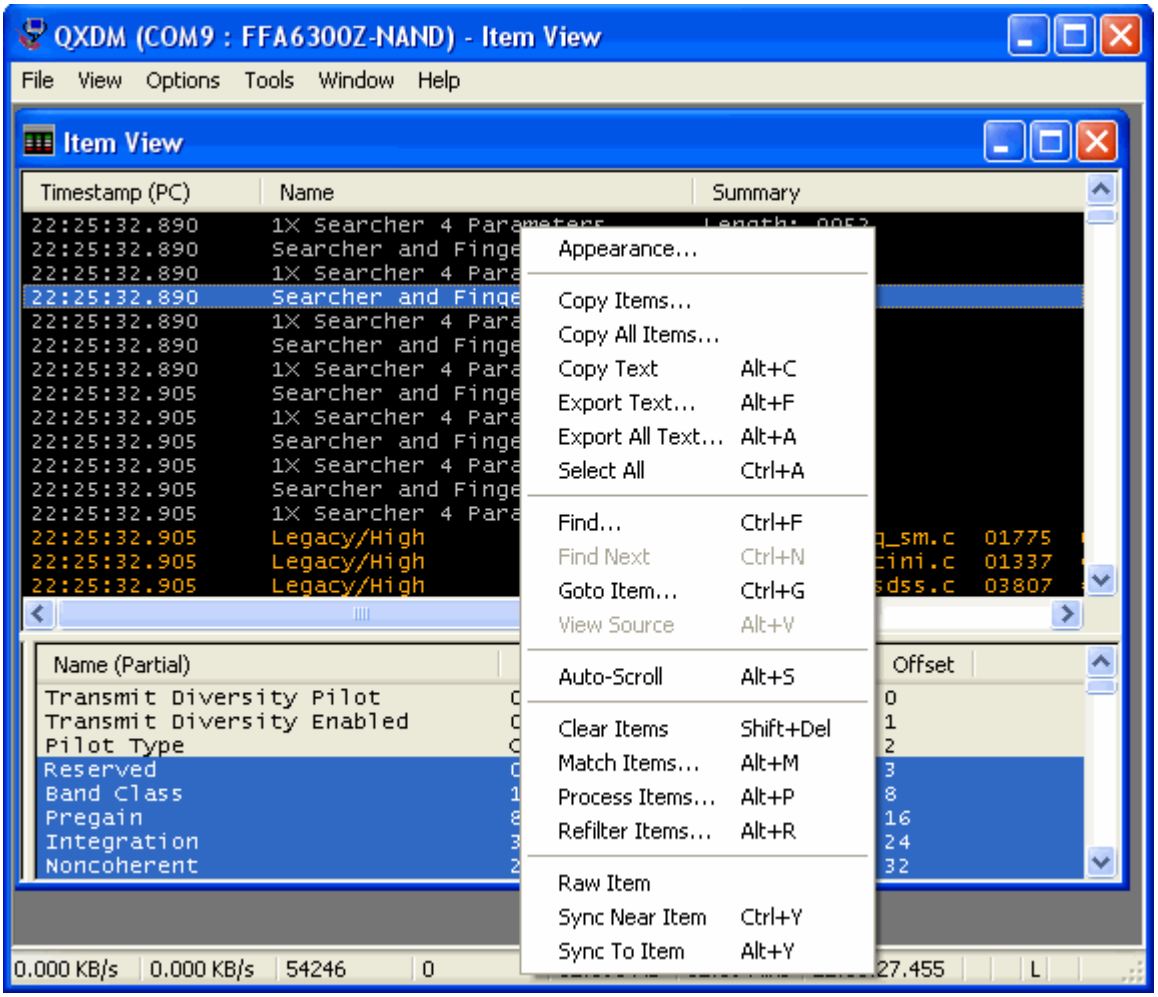

Figure 4-2 Context-sensitive menu for Scrolling List Pane

The context-sensitive menu is used to configure view registration, alter appearance, copy text, locate items, clear the view, and view raw item content.

4.2.1.2 Appearance

Selecting this menu option (Table 4-3) lets you control the appearance of the scrolling list by selecting which item properties to display (in the form of columns), by specifying the order in which to display item properties, and by configuring item property specific options.

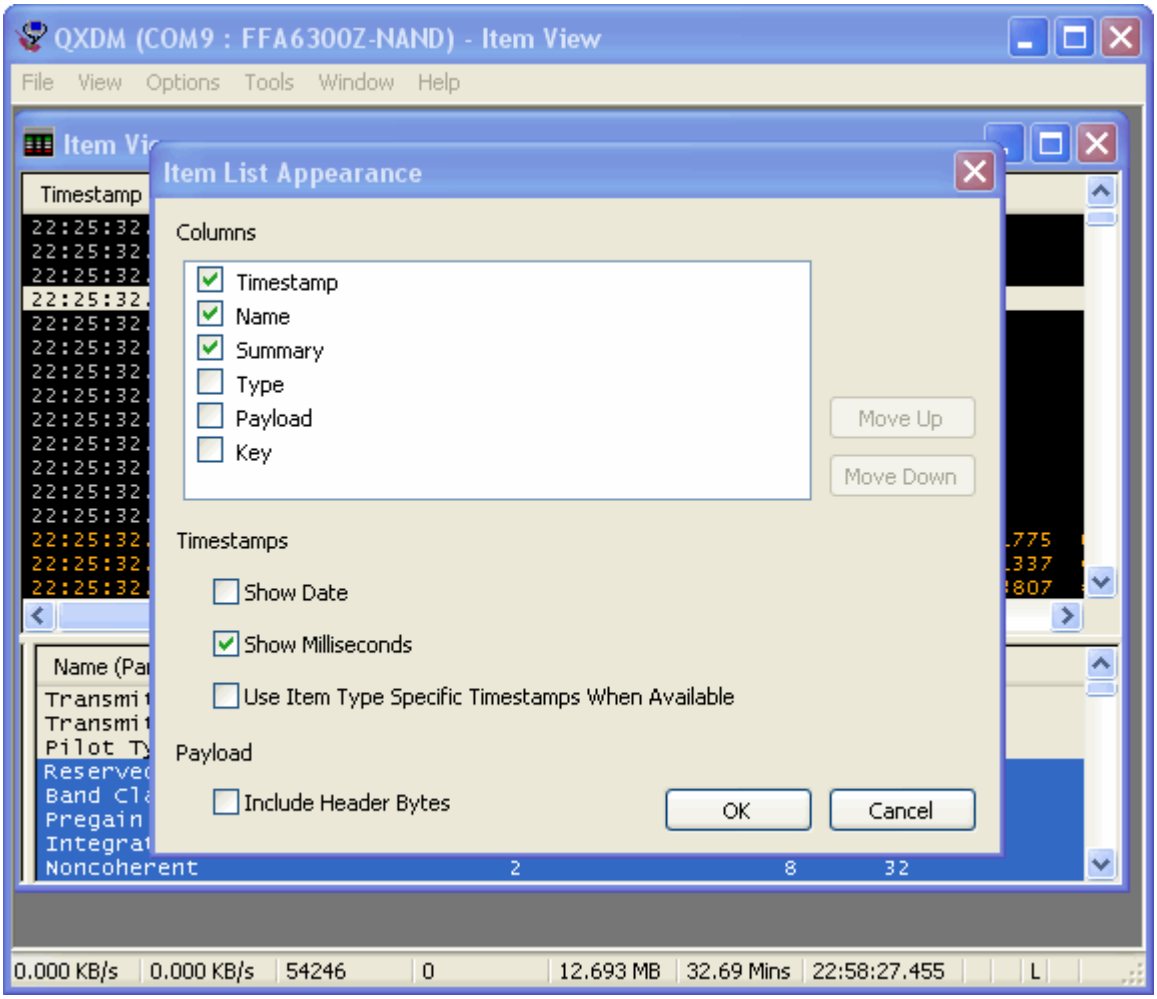

Figure 4-3 Item List Appearance

4.2.1.2.1 Timestamps

The timestamp is shown in Greenwich Mean Time (GMT) and can be modified to include date and milliseconds. PC time or when available item-specific time can also be configured. PC time is the PC system time adjusted to GMT, at which an item is added to the Item Store. Item-specific time is the timestamp included in the packet received from the phone. Items that include phone timestamps are logs, events, debug messages, and the timestamp response packet.

4.2.1.2.2 Payload

By default, only the payload of an item is displayed when the Payload column is selected. Check Include Header Bytes to see the entire item content.

4.2.1.3 Configuration

Selecting this menu option allows you to control which items are added to the view in question.

The left side of the configuration dialog (Figure 4-4) is used to identify what item types the filtered view will accept. The right side of the dialog is used to filter item type content based on item key and to register with the phone for items of interest.

NOTE This configuration dialog is only supported by Filtered Views. The legacy Messages View and Log View use view-specific configuration dialogs discussed in subsequent sections.

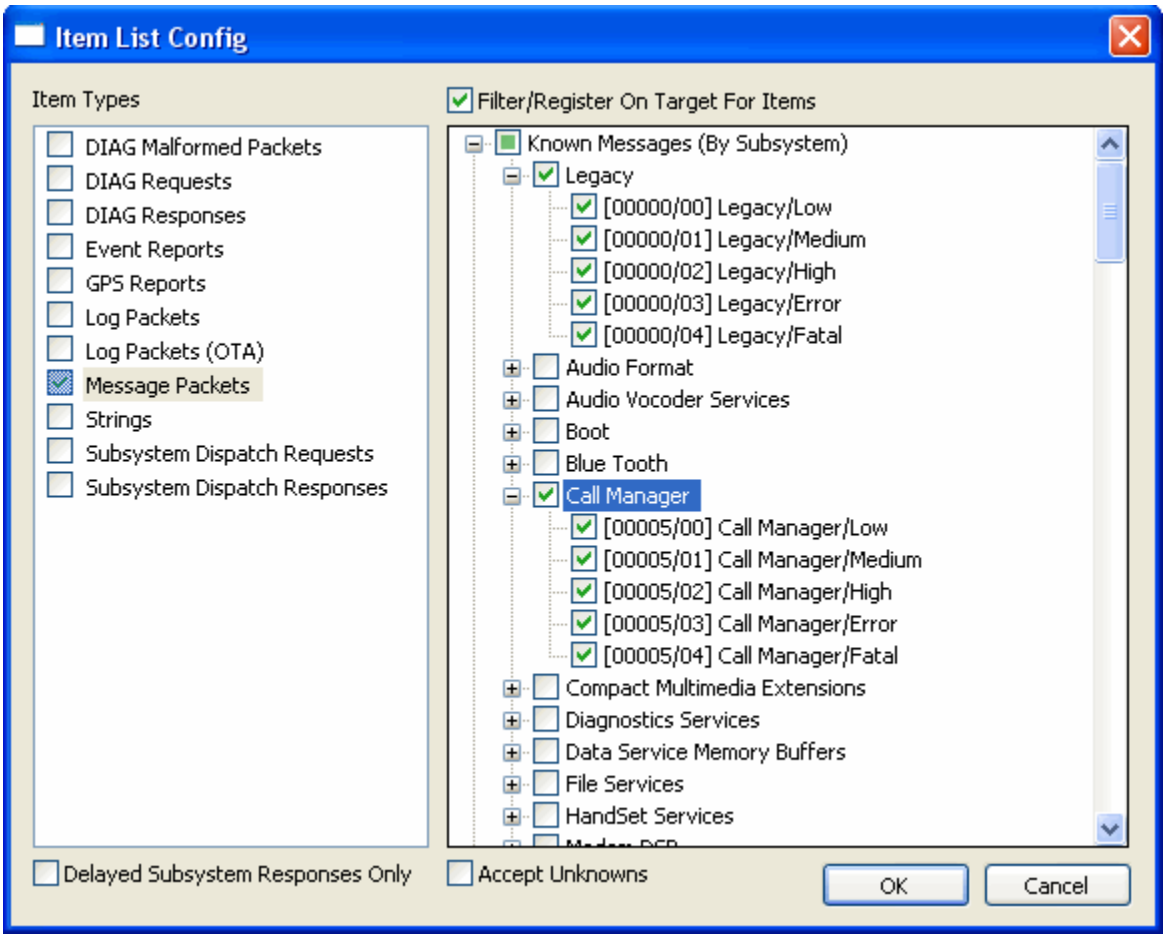

Figure 4-4 Item List Config dialog

Items are grouped into types, as shown in Table 4-1. Item types allow QXDM to classify incoming data from multiple sources: synchronous and asynchronous traffic between QXDM and a target, GPS receiver data, and internally generated strings.

Table 4-1 Item types

Item type	Source
DIAG Malformed Packets	Phone
DIAG Requests	QXDM, User
DIAG Responses	Phone
Event Reports	Phone
GPS Reports	GPS Receiver
Log Packets	Phone
Log Packets (OTA)	Phone
Message Packets	Phone
Strings	QXDM, User
Subsystem Dispatch Requests	QXDM, User
Subsystem Dispatch Responses	Phone

4.2.1.3.1 Delayed Subsystem Responses Only

Delayed Subsystem Responses Only, when selected, enables you to configure the view to receive only the delayed subsystem dispatch V2 responses (as opposed to both the immediate and delayed responses). Since both V2 and V1 responses are treated as the same item type, this makes certain types of filtering and item processing more convenient.

4.2.1.3.2 Accept Unknowns

Accept Unknowns, when selected, enables you to configure the view to receive items that are unrecognized. An item is considered unrecognized when no mapping between item type/key and item name exists in the QXDM database and any loaded user database.

4.2.1.4 Copying

All or part of a filtered view output can be copied to a new ISF, a text file, or to the clipboard. The action is controlled by using any of the methods given in Table 4-2.

Table 4-2 Copying Items

Item	Description
Copy Items	Copy selected items to an Item Store Format (.ISF) file
Copy All Items	Copy All Items to an Item Store Format (.ISF) file
Copy Text (ALT + C)	Copy selected text to the clipboard
Export Text (ALT + F)	Export selected items as text to a file
Export All Text (ALT + A)	Export all items as text to a file
Select All (CTRL + A)	Select all items

Text output from loaded dynamic parsers, built-in OTA parsers, and extended database descriptions will also be copied if the menu item Options → Export Parsed Text is selected (see Section 3.3.7).

NOTE Text output is subject to change. Use post-processing tools like QCAT to generate consistent text output for more reliable text processing results.

All of the above operations, excluding copying text to the clipboard, result in the display of a progress dialog. This dialog displays the progress of the operation, statistics related to the operation, and any errors encountered in the process of the operation (such as errors reading an item from the current ISF or writing an item out to the new ISF). Additionally, the progress dialog allows you to cancel the operation before it has completed. In this case, the partial results are written out to the new ISF or text file.

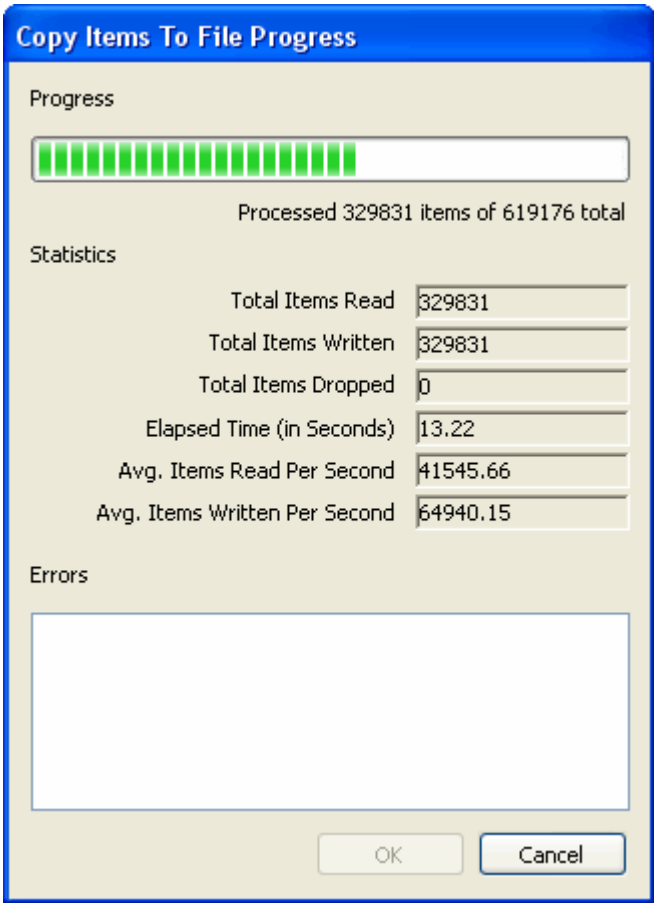

Figure 4-5 Copy To Items File Progress dialog

4.2.1.5 Searching

Each line in an item list view is the text representation of the properties of a particular item. This text can be searched in order to find an item that meets a given search criteria. Additionally each item in an item list view is associated with an index. This index can be used to jump to an item. For items with type Message Packets, additional information present in the content of the item can be used to locate the point in the target source code where the item originated.

4.2.1.5.1 Find

Selecting this menu option (or using the CTRL + F keyboard shortcut) allows you to search for text within the item list output. Both the type of item and the item properties to be searched can be configured.

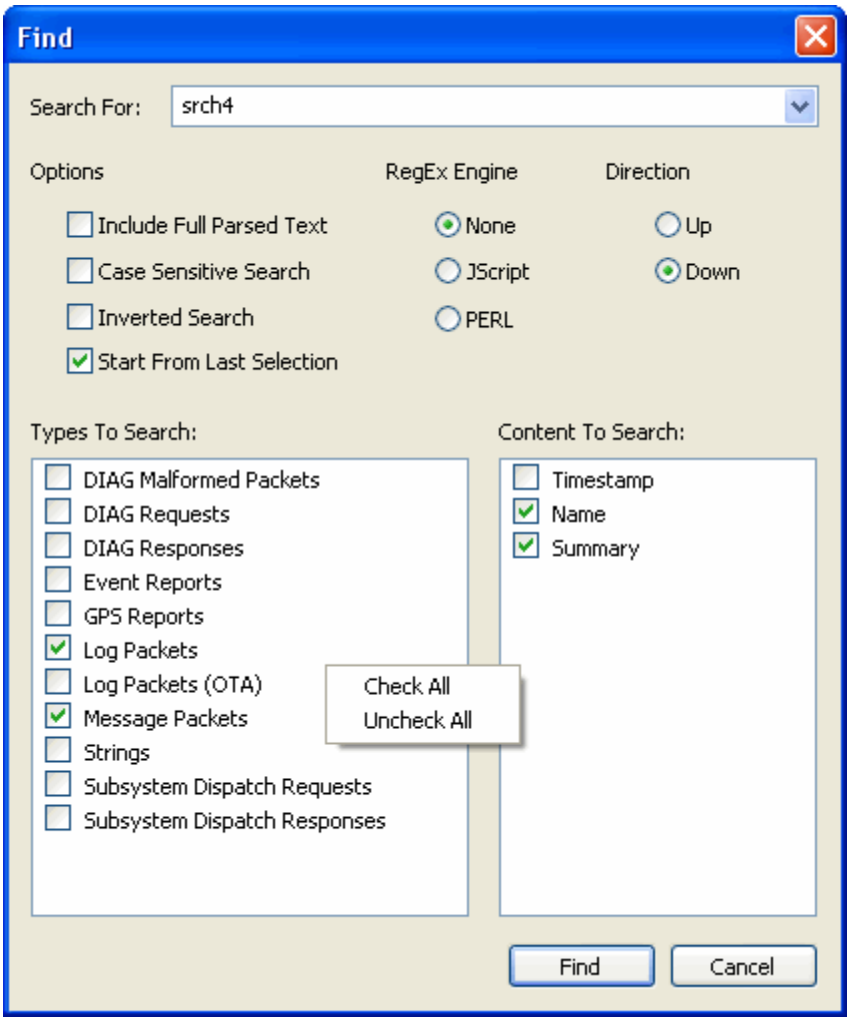

Figure 4-6 Find option for scrolling views

4.2.1.5.1.1 Find options

In addition to search direction, the following options are supported:

- Include Full Parsed Text (When Available) – When checked, the search engine will also search the full parsed text that appears in the lower right-hand pane of the view.
- RegEx Engine – The parser will accept regular expressions when JScript or Perl options are checked. Although support for JScript is included with the Windows operating system, Perl requires a separate installation. Documentation on regular expression syntax for these scripting languages is beyond the scope of this manual.
- Case Sensitive Search – By default, the search is case insensitive.
- Inverted Search – When checked, the search engine will search for any pattern not matching the search criteria from the set of item types and item properties.
- Start From Last Selection – By default, a new search is from the beginning of the file. Checking Start from Last Selection overrides this behavior.

4.2.1.5.2 Find Next

Find Next allows you to search for the next Find match. You can also execute this command by pressing CTRL + N.

4.2.1.5.3 Goto Item

You can go to an item by specifying its index using the Goto Item command. When successful, QXDM will highlight the item at the index specified. You can also execute this command by pressing CTRL + G.

4.2.1.5.4 View Source

View Source allows you to view the target source file associated with a selected item. You can also execute this command by pressing ALT + V. Use Options → Settings (see Section 3.3.2.3) to specify the search path used by QXDM.

4.2.1.6 Auto-Scroll

When new items arrive and are added to an item list view, QXDM can automatically scroll to the end of the item list in order to display the new items, The Auto-Scroll menu option allows you to enable or disable this behavior. You can also control automatic scrolling by pressing ALT + S or by manually scrolling using the scroll bar, arrows, or PAGE UP and PAGE DOWN keys.

4.2.1.7 Clear Items

Clear Items allows you to flush all items from an item list view. You can also clear items by pressing SHIFT + DELETE.

NOTE Items can be cleared from any scrolling view except the Logging <ALT + L> View. Take care when clearing the Item View because doing so results in the current ISF being emptied and the connection being reset.

4.2.1.8 Match Items…

Match Items allows you to create a new Filtered View based on a matched pattern from a selected set of items in an existing item list view. Configuring this process is nearly the same as configuring a find operation.

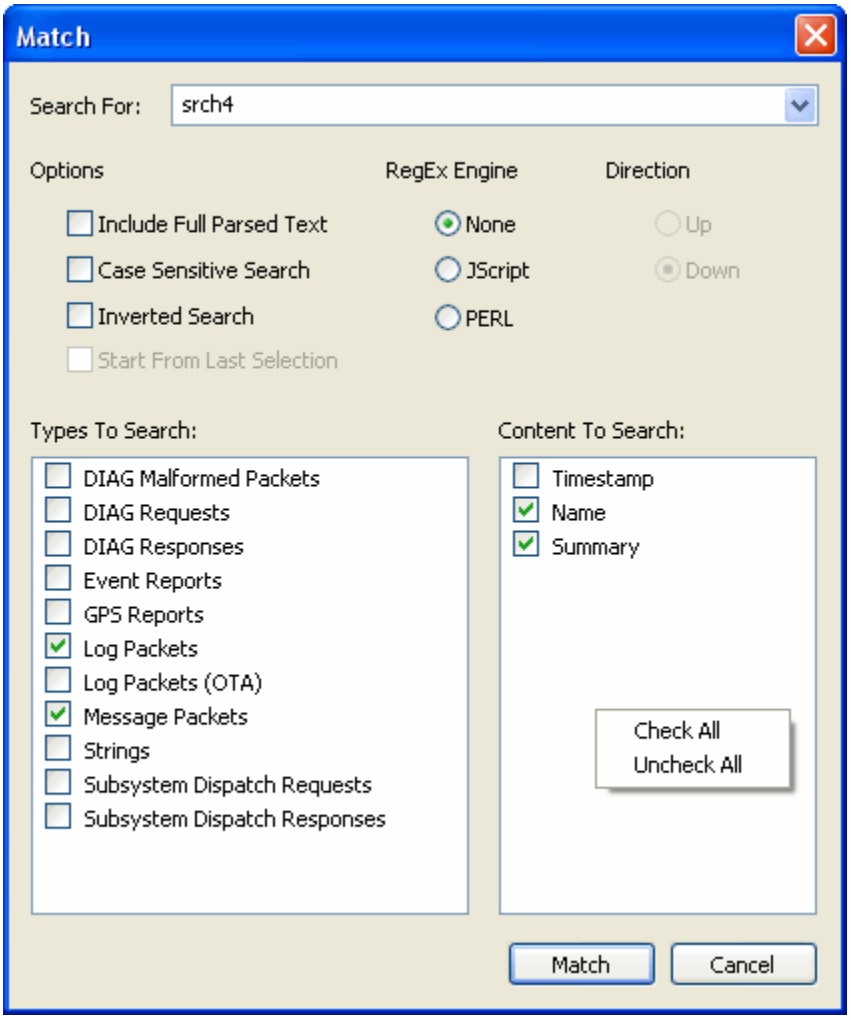

Figure 4-7 Match Items

The only difference is that the direction and selection options are not available. Instead the operation starts from the first selected item in the current item list view and ends with the last selected item.

4.2.1.9 Process Items

Process Items allows you to create a new Filtered View based on processing the selected items in an existing view with a user-provided script (Figure 4-8). The script can be written in any scripting language that supports the Microsoft Windows scripting engine interface model (IActiveScript). Examples of supported scripting languages are Perl, VB Script, and JScript. The contents of the user script must include a function named ProcessItem(), which accepts a single argument, an interface pointer to the QXDM item interface model (IColorItem) as shown in Table 4-3.

An example script, ProcessDelayedResponses.js, is provided in the QXDM Automation Samples folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\Automation Samples).

Table 4-3 ProcessItem() parameters

Name	Type	Description
Item	IColorItem	Refer to Section 8.4.14 for descriptions of supported commands for this interface

ProcessItem() should return a Boolean value (VARIANT_BOOL) where true (VARIANT_TRUE) indicates the item passed to the function should be included in the new filtered view and false (VARIANT_FALSE) indicates the item should be discarded.

NOTE The use of the VARIANT_BOOL COM type may not be directly supported in all scripting languages, in which case zero may be substituted for VARIANT_FALSE and a nonzero value may be substituted for VARIANT_TRUE.

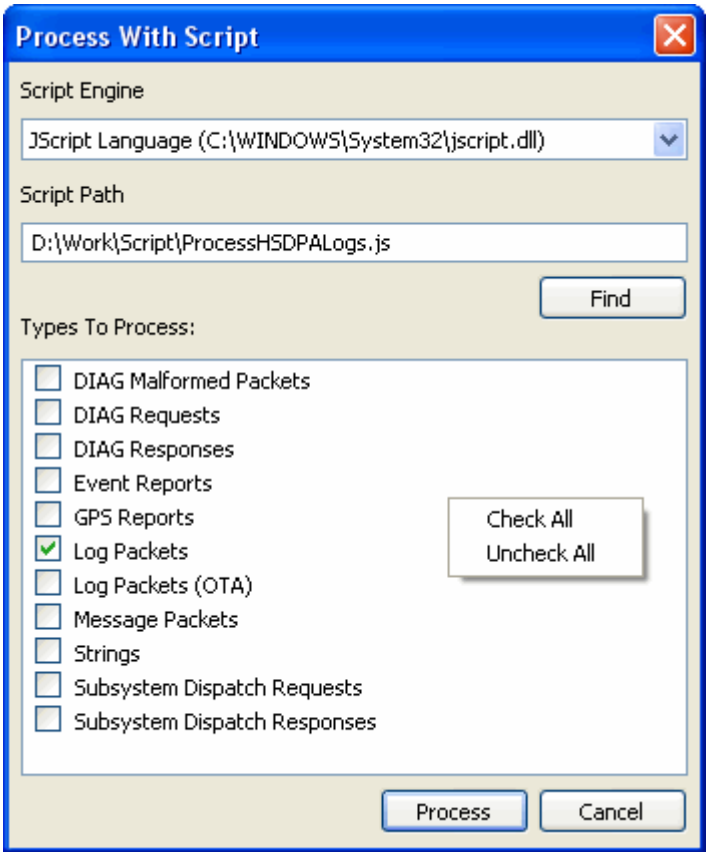

Figure 4-8 Process With Script

4.2.1.9.1 Process With Script options

The options are:

- Script Engine – Select the scripting engine to use from the available scripting hosts on your computer
- Script Path – Specify the script file that will do the IColorItem processing
- Types To Process – Select the item types your script is interested in processing

4.2.1.10 Refilter Items

Refilter Items allows you to create a new Filtered View from a selected group of items in an existing view based on item type and/or item key. Configuring the refilter operation is no different than configuring a Filtered View.

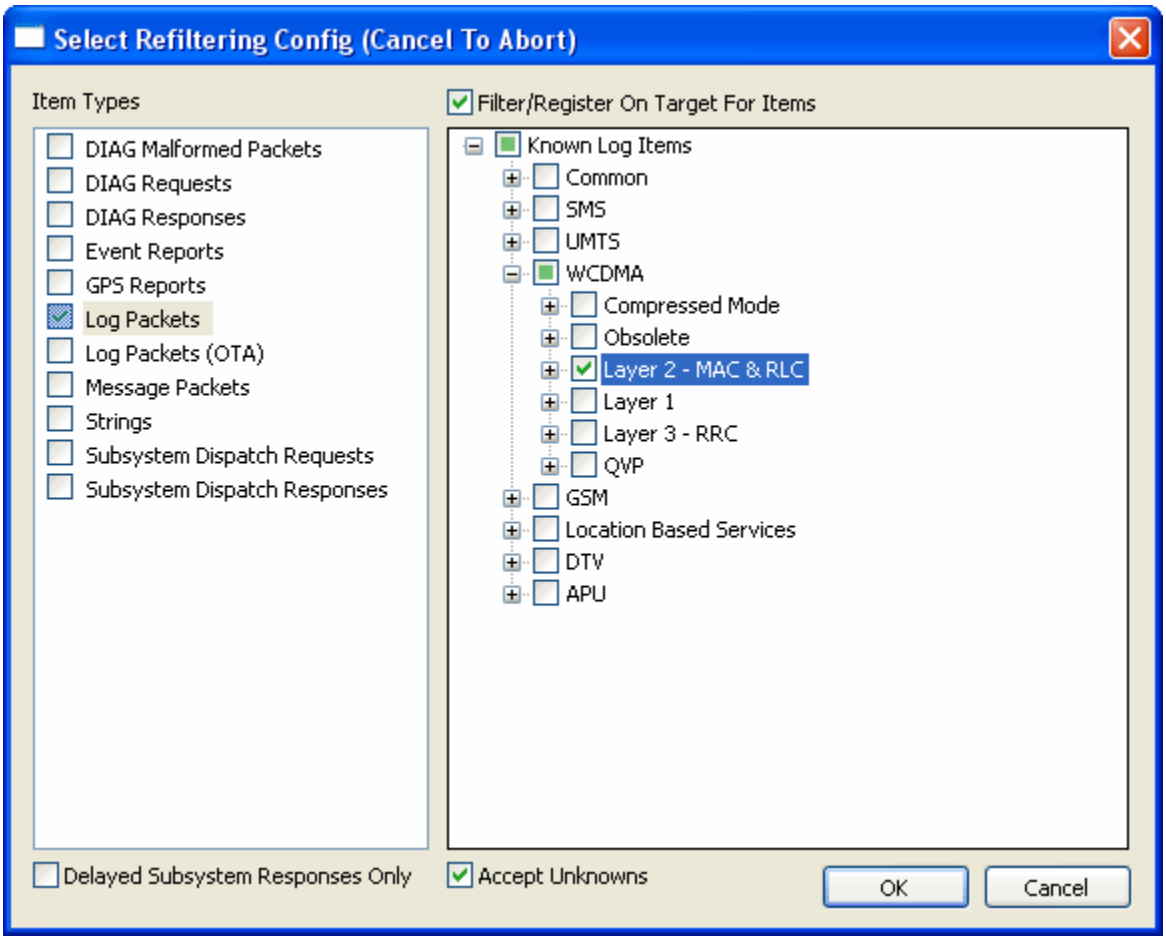

Figure 4-9 Refilter Items

4.2.1.11 Raw Item

This option displays the raw item contents in hexadecimal form for the currently selected item. A context menu can also be used to control the layout by clicking the right mouse button over the Raw Item Contents view.

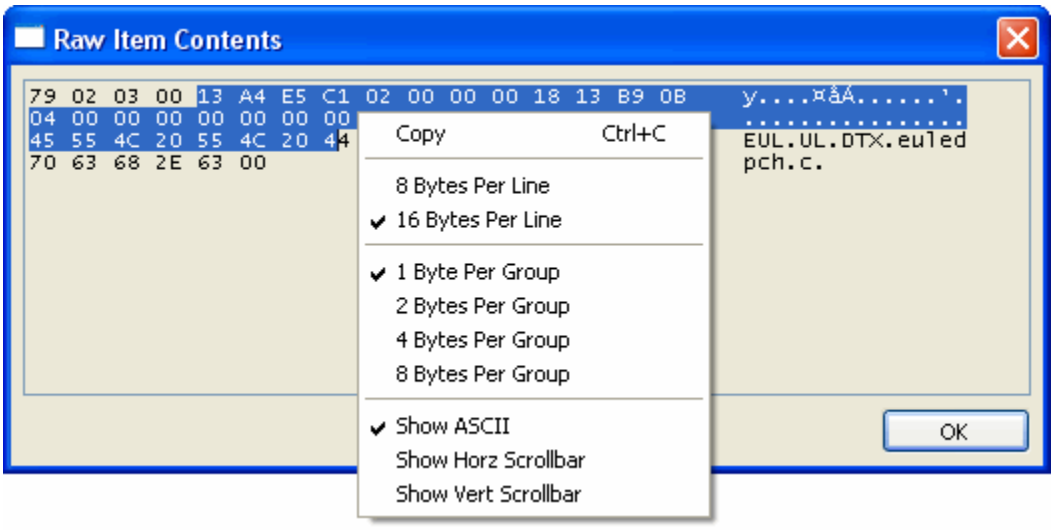

Figure 4-10 Raw Item menu

4.2.1.12 Sync Near Item

Sync Near Item tries to find the item with the given index in all item lists. In each list, if the item with the given index is found, it is selected. If the item with the given index is not found, the item with the index closest to the specified index is selected.

NOTE It should be noted that item position correlates to generic timestamp, i.e., the item at index N has a generic timestamp that is earlier than the item at index N + 1, etc. In general, there is no correlation between item position and item-specific timestamps as they originate external to QXDM.

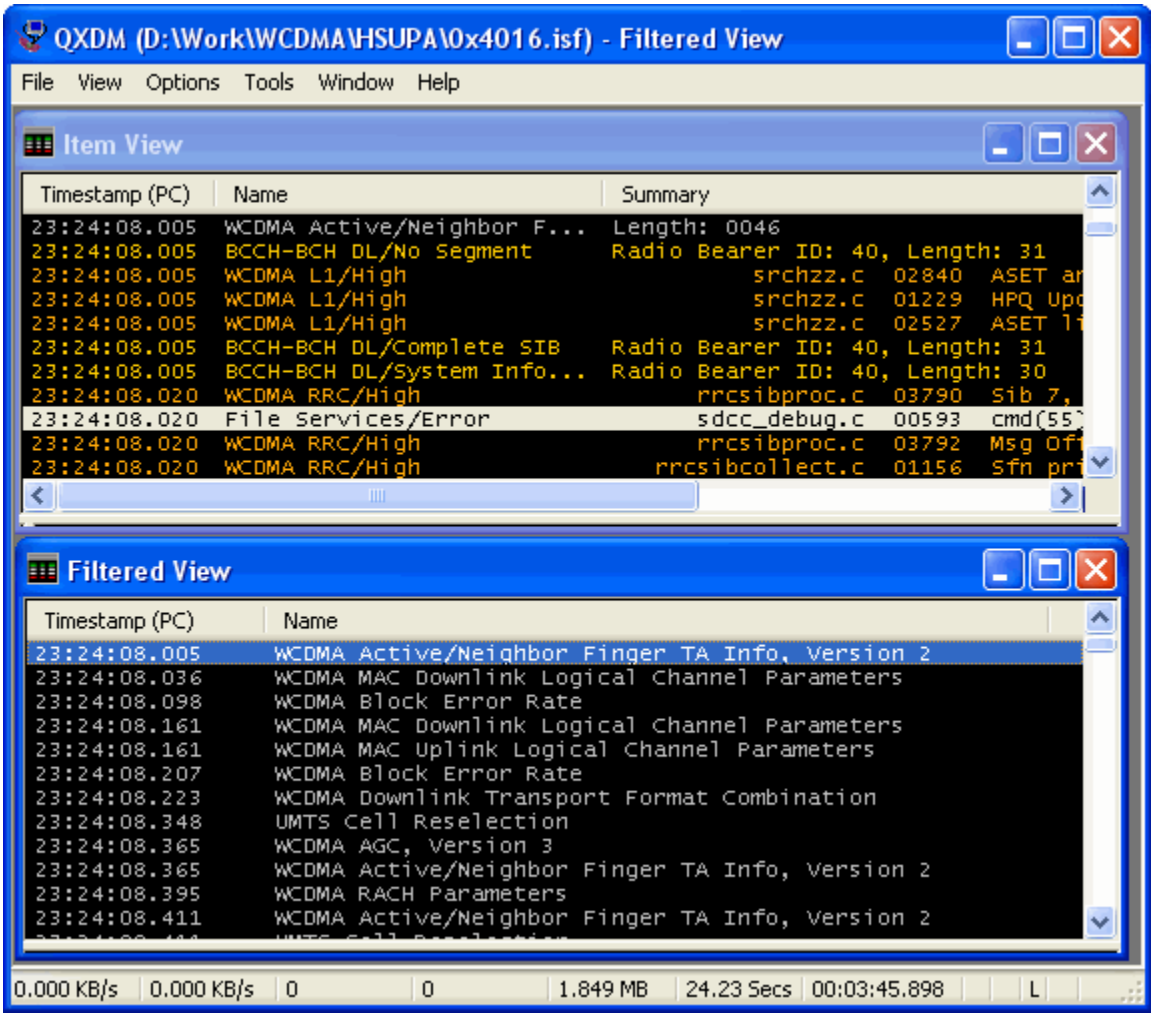

Figure 4-11 Sync Near Item

4.2.1.13 Sync To Item

Sync To Item syncs all filtered views to the same selected item, if found. Sync To Item tries to find the item with the given index in all item lists. In each list, if the item with the given index is found, it is selected. If the item with the given index is not found, no change is made.

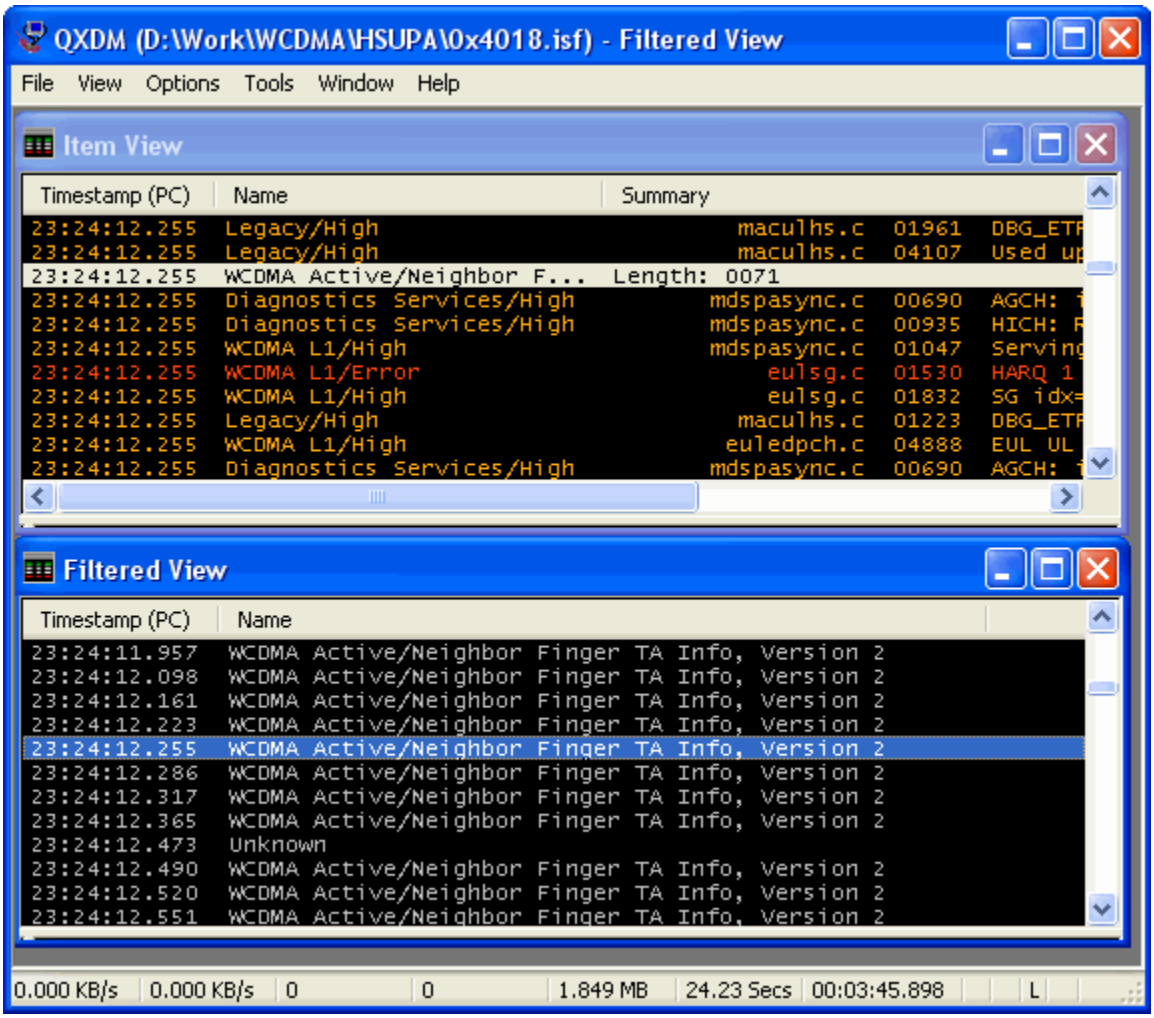

Figure 4-12 Sync To Item

4.2.2 Raw Item Pane

The Item View and Filtered Views contain the Raw Item Pane. This pane displays the raw item contents in hexadecimal form for the item currently selected in the Scrolling List Pane.

4.2.2.1 Menu options

Right-clicking within the Raw Item Pane brings up the context-sensitive menu.

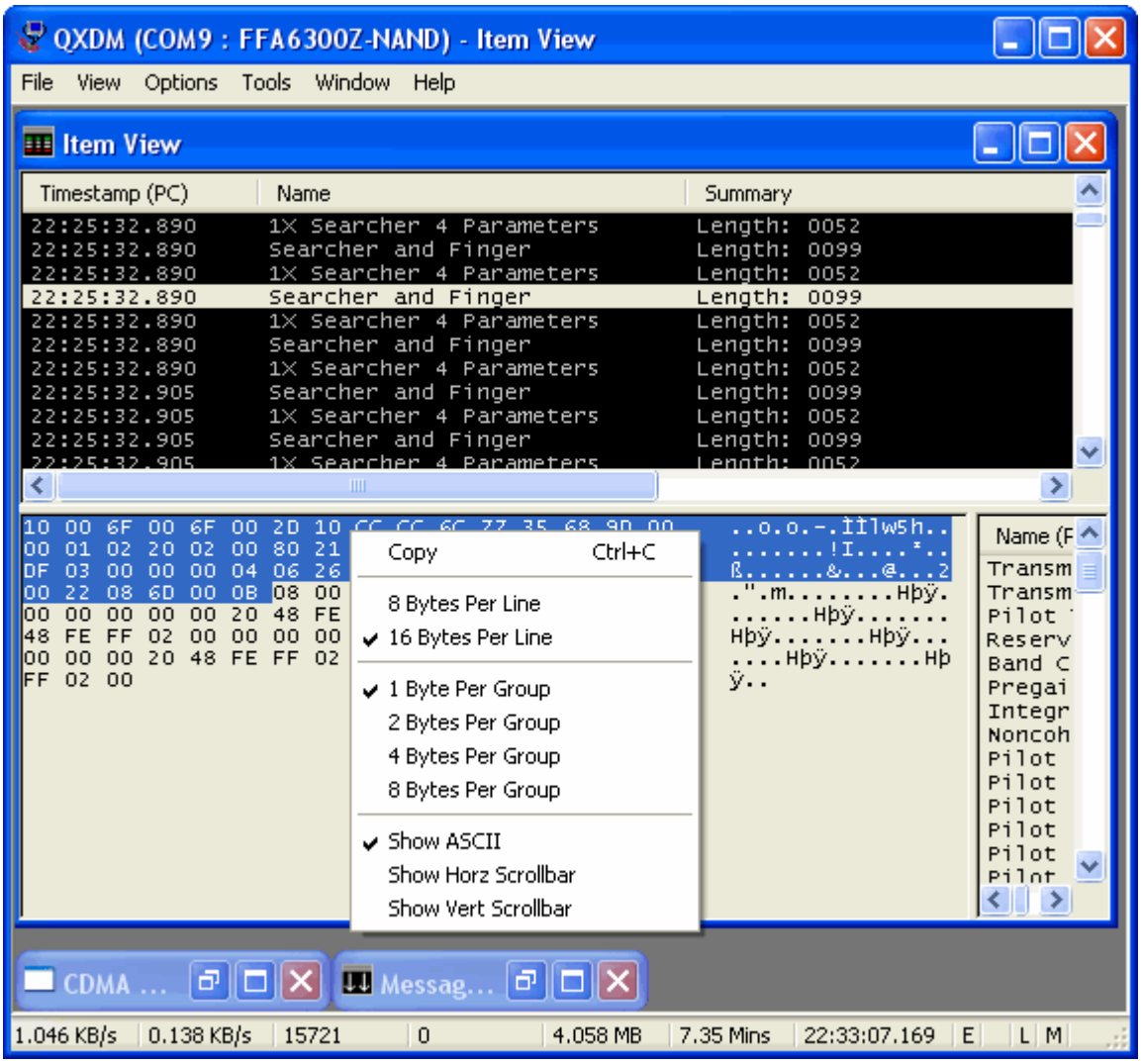

Figure 4-13 Raw Item Pane menu

4.2.3 Parsed Item Pane

The Item View and Filtered Views contain the Parsed Item Pane. This pane displays the parsed fields or, when available, the parsed text of the currently selected item in the scrolling list pane.

NOTE Not all items have associated parsers to display parsed text in the parsed item pane. Item data is parsed using definitions in the QXDM database, SILK, and ASN.1 for OTA log items, or dynamic link parsers (DLLs) written by end users (see Section 3.1.1).

4.2.3.1 Menu options

Right-clicking within the Parsed Item Pane brings up the context-sensitive menu.

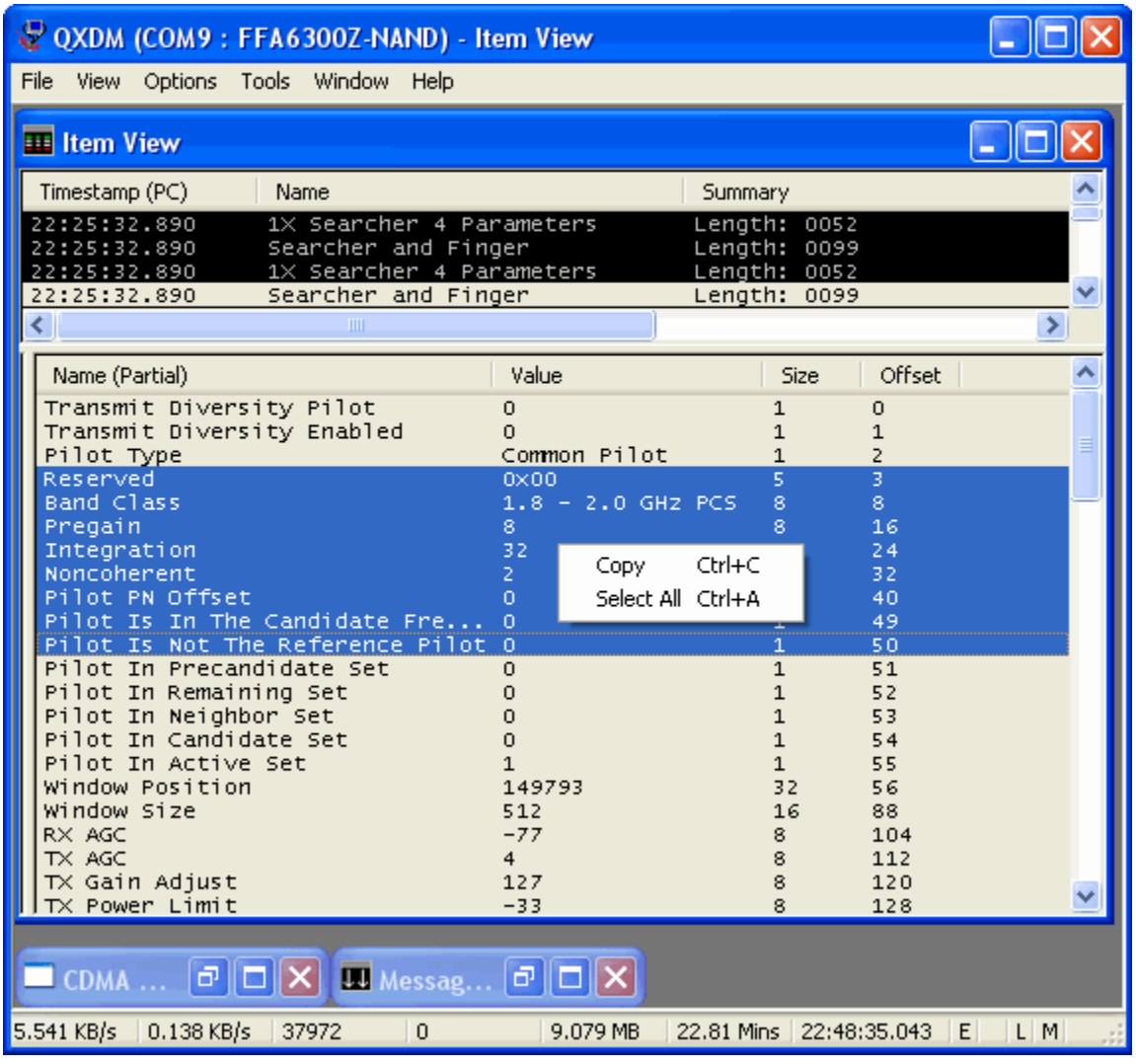

Figure 4-14 Parsed Item Pane menu

4.3 HTML views

QXDM provides an IE browser framework to display HTML views that process and display data received from the connected SURF or FFA. This framework utilizes the QXDM COM automation interface.

NOTE The IQXDM interface is only intended for use with the scripting interfaces and is not supported in the HTML views. Use the IQXDM2 interface for HTML view development (see Section 8.3).

HTML views are made accessible from the QXDM View Bar (Figure 3-21) when the following conditions are met:

- The file must have the extension .HTML and be located in the QXDM HTML folder (typically installed at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\HTML).
- The file must contain the following meta elements defined within the head element:
  - `DMViewName`
  - `DMViewWidth`
  - `DMViewHeight`

Existing QXDM views that demonstrate this format can be found in the QXDM HTML folder:

```
<!doctype html public "-//w3c//dtd xhtml 1.0 strict//en"
"http://www.w3.org/tr/xhtml1/dtd/xhtml1-strict.dtd">
<html>
<head>
   <meta name="DMViewName" content="Your HTML View Name" />
   <meta name="DMViewWidth" content="600" />
   <meta name="DMViewHeight" content="400" />

   <title>Your HTML View Name</title>
   <link rel="stylesheet" href="QXDMStyle.css" />
</head>
```

NOTE A default cascading style sheet file (QXDMStyle.css) is installed in the QXDM HTML folder. This file controls the font faces, font sizes, table layout, and overall color scheme used by all native QXDM HTML displays. Although you can override these settings by editing the contents of this file (the discussion of which is beyond the scope of this document), the changes will be overwritten the next time QXDM is upgraded.

Using the QXDM COM interface, you can write your own custom HTML views that process and display data received from the target. Refer to the QXDM HTML folder for various examples of HTML views. Refer to Section 8.3 for a complete description of supported methods and properties.

4.3.1 Menu options

Right-clicking within any HTML view brings up a context-sensitive menu, as illustrated in Figure 4-15.

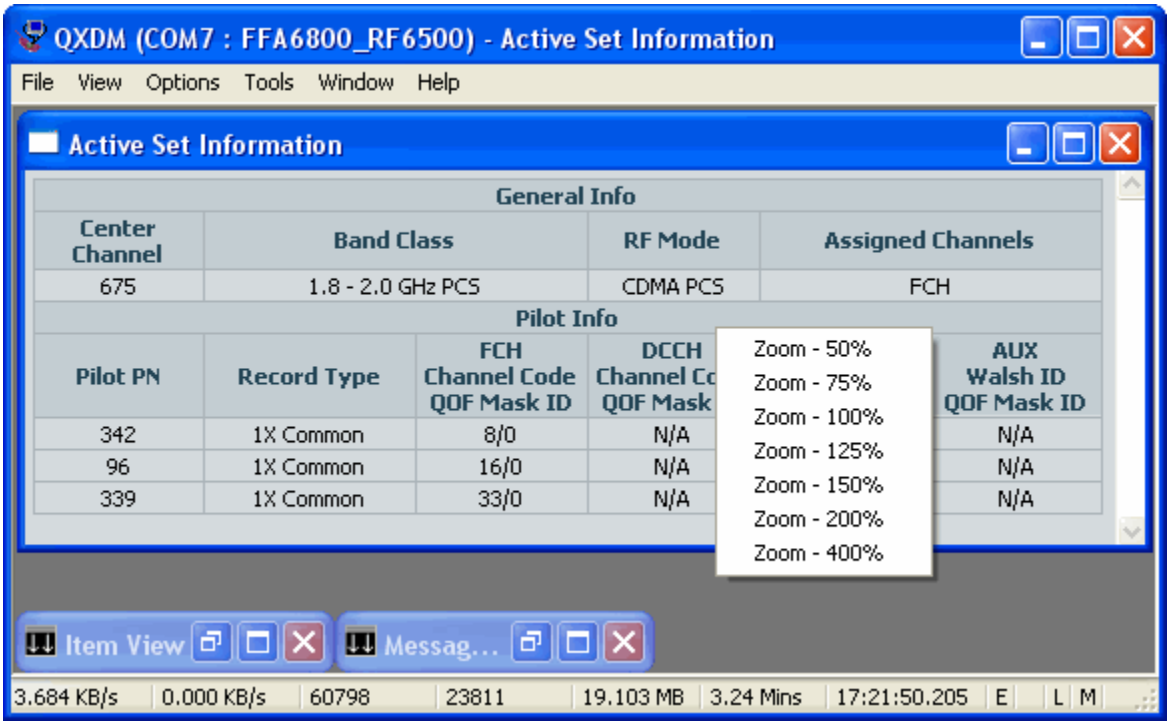

Figure 4-15 HTML view menu

This menu allows you to change the zoom level of the view. The default zoom level is 100%. Decreasing the zoom level decreases the size of the view contents. Increasing the zoom level increases the size of the view contents.

4.4 Graph views

Graph views exist for the technologies listed in Table 4-4.

Table 4-4 Graph views

View name	Technology
CDMA Temporal Analyzer	1X
CDMA Finger Placement	1X
CDMA Link Info - Forward	1X
CDMA Link Info - Reverse	1X
CDMA Power	1X
CDMA Reverse Link	1X
CDMA RLP Throughput	1X
GSM GPRS Air Interface Summary	GSM/GPRS
GSM GPRS Tx Timing	GSM/GPRS
HDR Air Link Summary	1xEV-DO
HDR Power	1xEV-DO
HDR Rev A DRC-DSC-ARQ Buffer Metrics	1xEV-DO
HDR Rev A Equalizer Metrics	1xEV-DO
HDR Rev A Forward Link Packet Header Info	1xEV-DO
HDR Rev A Forward Link User Packet Throughput	1xEV-DO
HDR Rev A RAB Info	1xEV-DO
HDR Rev A Reverse Link T2P Statistics	1xEV-DO
HDR Rev A Reverse Link Gain Ratios	1xEV-DO
HDR Rev A Reverse Link Packet Info	1xEV-DO
HDR Rev A Reverse Link Packet Status	1xEV-DO
HDR Rev A Reverse Link Throughput	1xEV-DO
HDR Temporal Analyzer	1xEV-DO
Task Profiling	Common
Qtv™ Audio Video Sync	Common
Qtv Buffer Occupancy	Common
Qtv Inter-Arrival Jitter	Common
Qtv Packets Lost	Common
Qtv Streaming RX Statistics	Common
UMTS Cell Reselection	UMTS
VoIP QDJ Parameters	Common
VoIP QDTX Parameters	Common
WCDMA Active Set PSC Strength	WCDMA/HSDPA
WCDMA BLER	WCDMA
WCDMA BLER Total	WCDMA
WCDMA Capacity/Quality Measurements	WCDMA

View name	Technology
WCDMA CM GSM Measurement	WCDMA
WCDMA HSDPA Decoding Statistics	WCDMA/HSDPA
WCDMA HSDPA Link Statistics	WCDMA/HSDPA
WCDMA Pilot Scanner	WCDMA
WCDMA Power	WCDMA
WCDMA Power Control	WCDMA
WCDMA Radio Bearer Rate	WCDMA
WCDMA Temporal Analyzer	WCDMA/HSDPA

Graph views provide various configurations that are accessible from context menus. Pressing the right mouse button while over the view brings up the context menu allowing various different configurations.

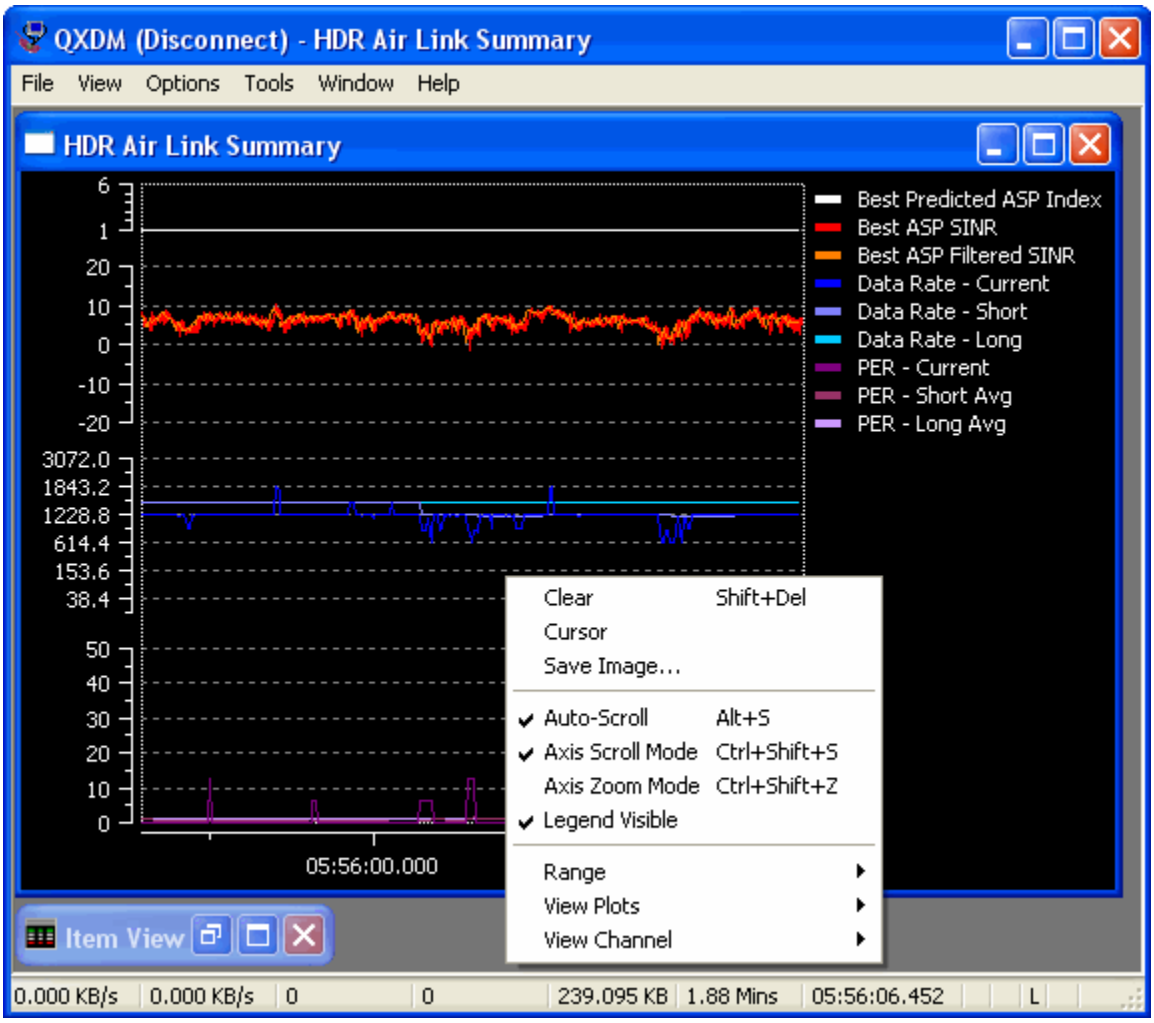

Figure 4-16 Graph view context menus

4.4.1 Clear

Selecting Clear causes the history of all information plotted to be cleared.

4.4.2 Cursor

Selecting Cursor causes a vertical line cursor to displayed or hidden on views that support this feature. Values for items over which the cursor is placed are displayed in the legend. The cursor can be dragged with the mouse to highlight points on the graph.

4.4.3 Save Image

Use Save Image to save the graph display contents to an image file in the BMP, JPG, or PNG image file formats.

4.4.4 Auto-Scroll

Selecting Auto-Scroll toggles automatic scrolling of the graph on views that support this feature.

4.4.5 Axis Scroll Mode

Selecting Axis Scroll Mode enables scrolling of the X or Y axis on views that support this feature. Holding down the left mouse button over the X or Y axis and dragging allows you to scroll.

4.4.6 Axis Zoom Mode

Selecting Axis Zoom Mode enables zooming of the X or Y axis on views that support this feature. Holding down the left mouse button over the X or Y axis and dragging allows you to control the range of viewable data in the window.

4.4.7 Legend Visible

Selecting Legend Visible toggles the display of the legend on views that support this feature.

4.4.8 Range

Use the Range option to select from a list of default zoom settings on views that support this feature.

4.4.9 View Plots

Use the View Plots option to select from a list of available plots on views that support this feature.

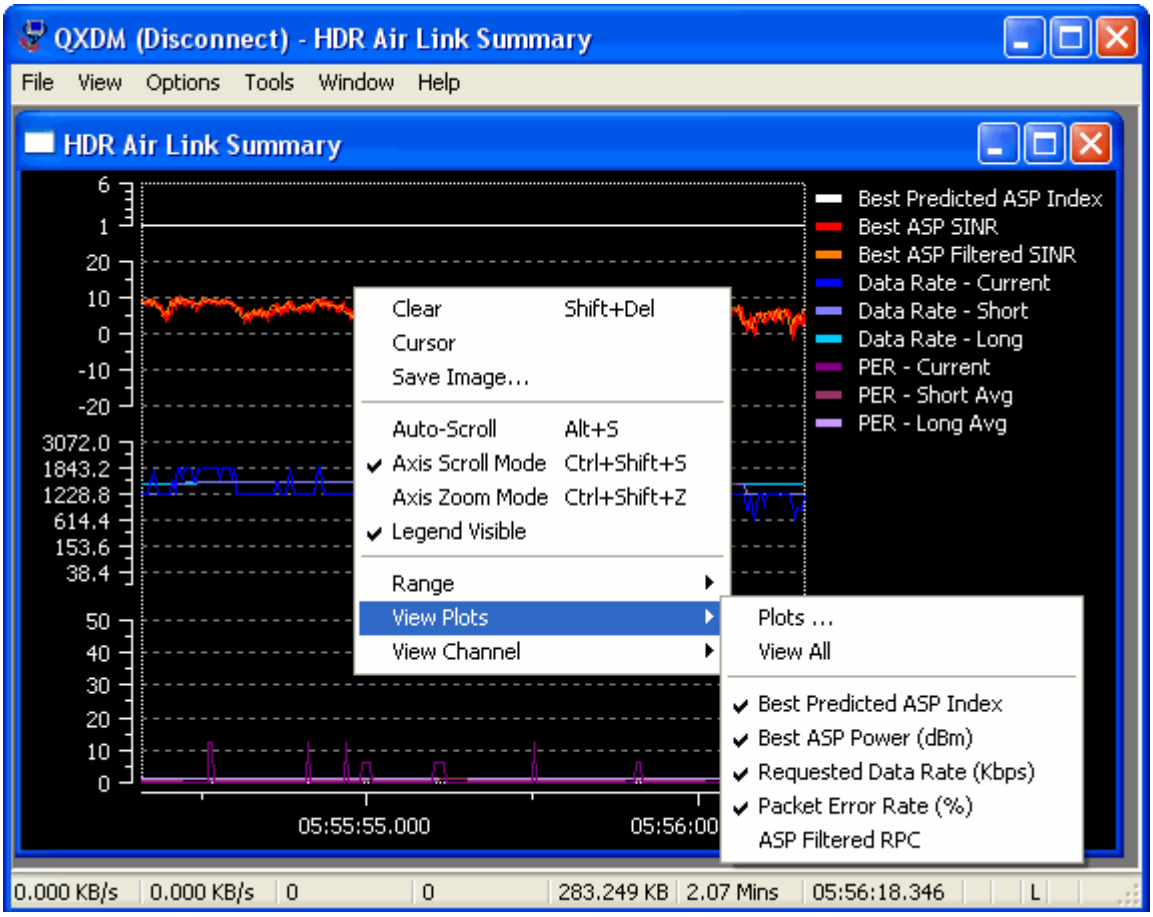

Figure 4-17 View Plots context menu option

Plots generally correspond to individual Y-axes.

4.4.10 View Channel

Use the View Channel option to select from a list of available channels on views that support this feature.

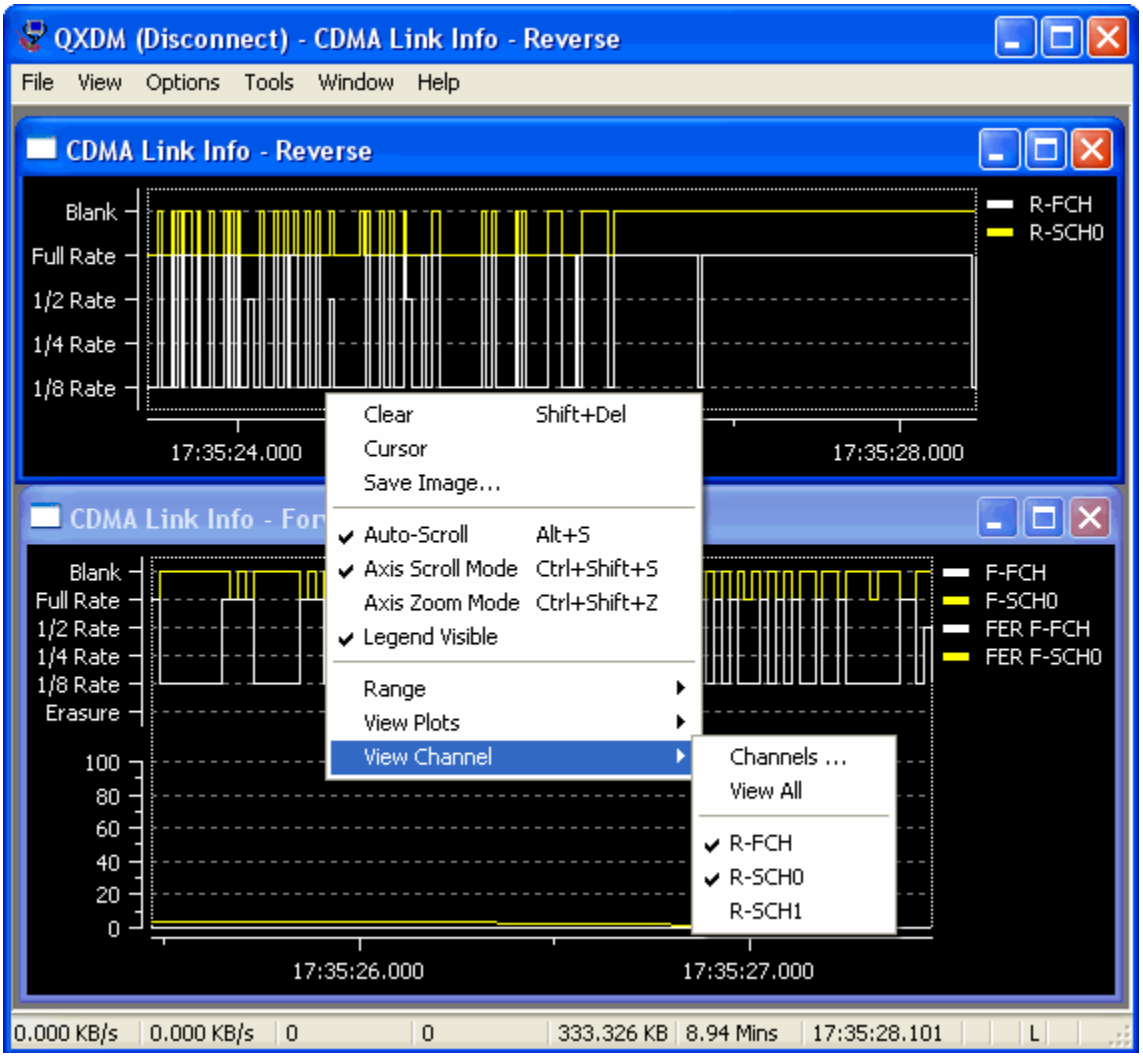

Figure 4-18 View Channel context menu option

4.5 Other views

4.5.1 Application Statistics

The Application Statistics view provides detailed information relating to item traffic, item totals, and the Item Store. The display is broken into three sections.

4.5.1.1 Communications Port

Communications statistics are accumulated since application startup.

4.5.1.2 Item Totals

Item Totals are for the current connection and all connections. Statistics are accumulated for items added to the Item Store, not items from a loaded ISF (File → Load Items). These totals are reset when the current Item Store is cleared or replaced.

4.5.1.3 Item Store

These statistics refer to the status of the current ISF, either loaded or temporary, and a history of the final statistics of each ISF file saved by QXDM (up to 100 of the last saved ISF files are displayed).

4.5.2 Dynamic Item View

OTA log items, items described in either the QXDM database or a user database, or items registered for by currently loaded dynamic runtime parsing DLLs, can be viewed periodically using the Dynamic Item View. Right-click within the Dynamic Item View to configure what item to view periodically (Figure 4-19).

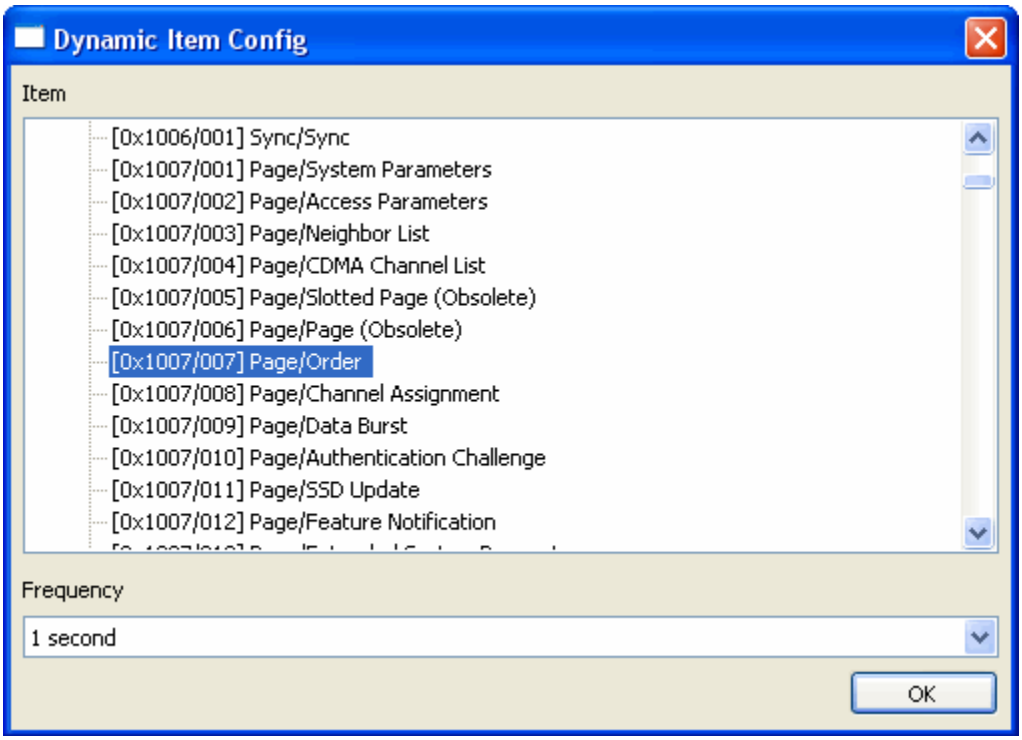

Figure 4-19 Dynamic Item Config

The frequency at which the view is refreshed can be configured from once every 50 ms to once every 5 sec. Only one item can be selected per view. However, any number of Dynamic Item Views can be created (Figure 4-20).

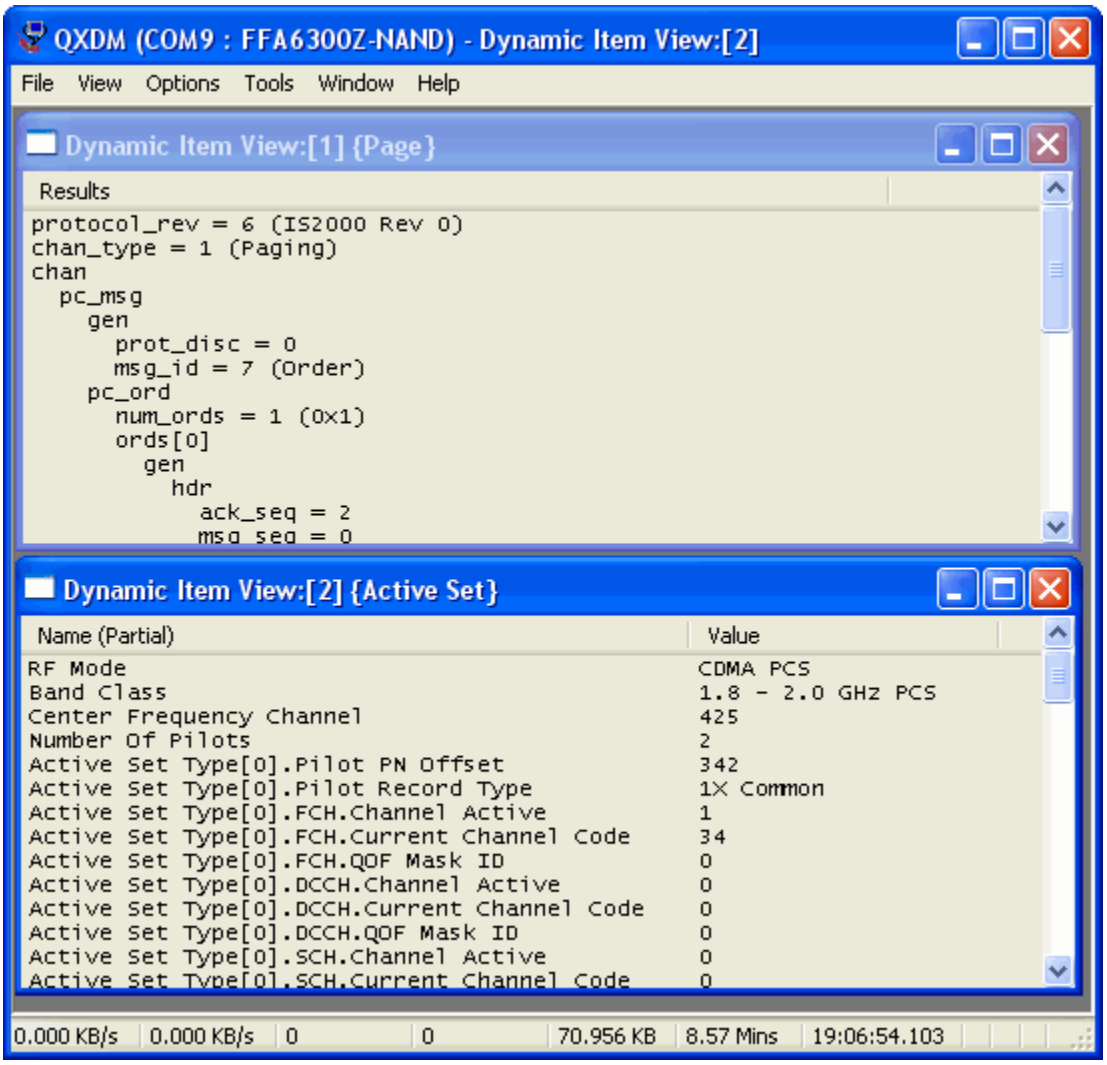

Figure 4-20 Dynamic Item Views

4.5.3 NV Browser

NV items stored in the nonvolatile memory of the connected target can be viewed and modified using the NV Browser.

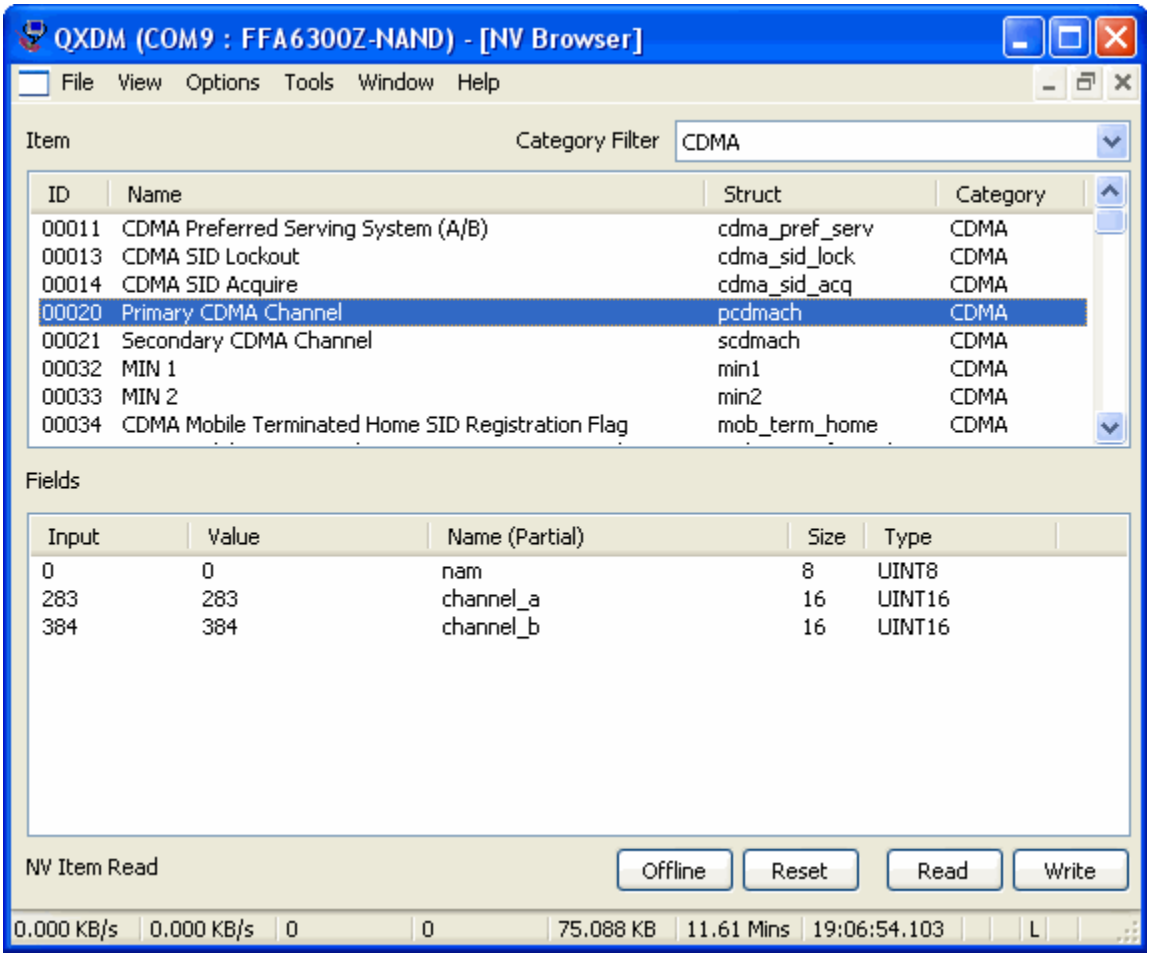

Figure 4-21 NV Browser

The Category Filter can be used to view just NV items belonging to a single category.

To read an NV item, select the item to display the names of all the fields for that item and click the READ button to read the values from the phone. To write an NV item, select the item to display the names of all the item fields. To change the values, click the value in the Input column. After modifying the values, click the WRITE button to write the updated values back to the phone.

NV items can be sorted by clicking the column header. Columns can be reordered by holding down the left mouse over a column header and dragging it to a new location. The left-most column can be searched by typing the name or number of interest.

NOTE Search typing is case-sensitive.

The status of the NV read and write is given in the bottom left portion of the display. If you click READ or WRITE and see DIAG Error Received or NV Status Error Received, then the target was not able to handle the request.

4.5.3.1 Offline

Use the Offline button to send the Mode Change Request Offline command to the phone before writing NV items that have mode-specific requirements.

4.5.3.2 Reset

Use the Reset button to send the Mode Change Request Reset command to the phone. This command when sent to a phone in Offline mode will result in the phone being recycled.

4.5.3.3 Read

Use the Read button to read the selected NV item from the NV memory of the connected target.

4.5.3.4 Write

Use the Write button to write the selected NV item to the NV memory of the connected target.

4.5.4 Keypad

The keypad provides a simple key display that simulates the handset keypad and display. It currently only supports a text-based display.

4.5.5 Memory Viewer <F4>

This display allows you to view and edit memory locations inside the phone at runtime. Multiple memory locations can be viewed simultaneously by bringing up multiple instances of this view.

By default, the left side of the view shows the memory address for each line, the middle shows the memory contents in hexadecimal, and the right side shows the ASCII character values of the memory. Memory that is readable is displayed as black characters and can be edited. If the memory is not readable, it will be grayed out. To edit memory, modify the memory contents by clicking in the region and typing new values. Memory values that have been modified for writes are displayed in red.

NOTE Bytes displayed in red are buffered changes that have not yet been sent to the target, and the contents of memory on the target are not changed until WRITE is clicked. Note also that WRITE is disabled until there are buffered changes to write.

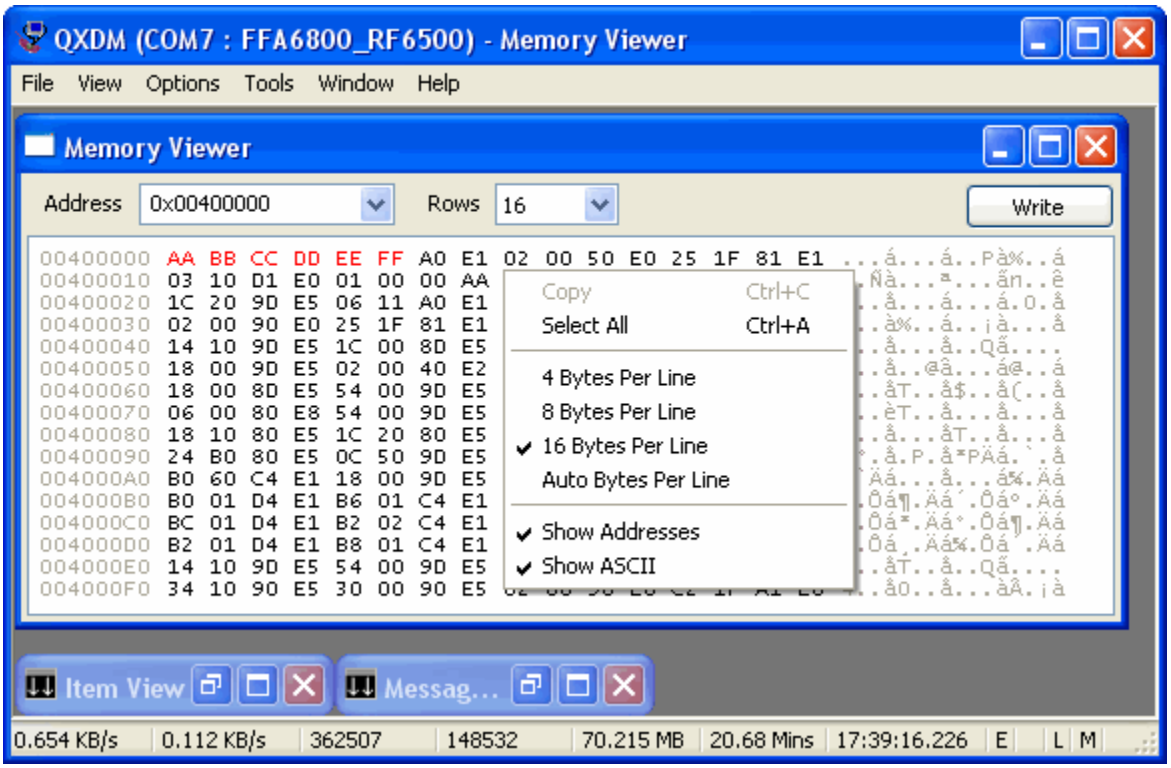

Figure 4-22 Memory Viewer context menu

4.5.5.1 Address

To view other locations in memory, modify the address with a new address by selecting from the available addresses in the Address combo box or by typing a new address in the Address combo box and then pressing the ENTER key.

4.5.5.2 Rows

The value specified in the Rows edit box determines the number of rows that are displayed. To change the value, select from the available list in the combo box.

NOTE Each row results in one DIAG request being sent to the target with a frequency of 1 sec, so the fewer the number of rows, the lesser the burden on DIAG.

4.5.5.3 Write button

The Write button will be grayed out if the memory address is unreadable. Clicking WRITE will result in a memory write request being sent to the phone.

4.5.5.4 Context menu

Use the context menu by clicking the right mouse button when held over the Memory Viewer. Control the appearance by hiding or displaying the addresses and ASCII characters and by controlling the number of bytes per line to display. Memory text can also be copied to the clipboard for pasting into another application.

4.5.6 OS Core Dump

By default, the OS Core Dump View is automatically launched when QXDM is started. This view registers interest in the OS Core Dump log. If a target that supports this log experiences failure, it attempts to send this log. Upon receipt, the log contents are displayed in this view. To disable OS Core Dump View registration, uncheck Auto-Launch Core Dump View from the Options menu.

4.5.7 Task Profiling

The Task Profiling view displays the results of the System-level Task and Per-task profiling logs. Per-task information is listed in the Task List pane and can be sorted in row order by clicking column headers. Tasks can also be plotted versus time in the Task Graph pane of this display.

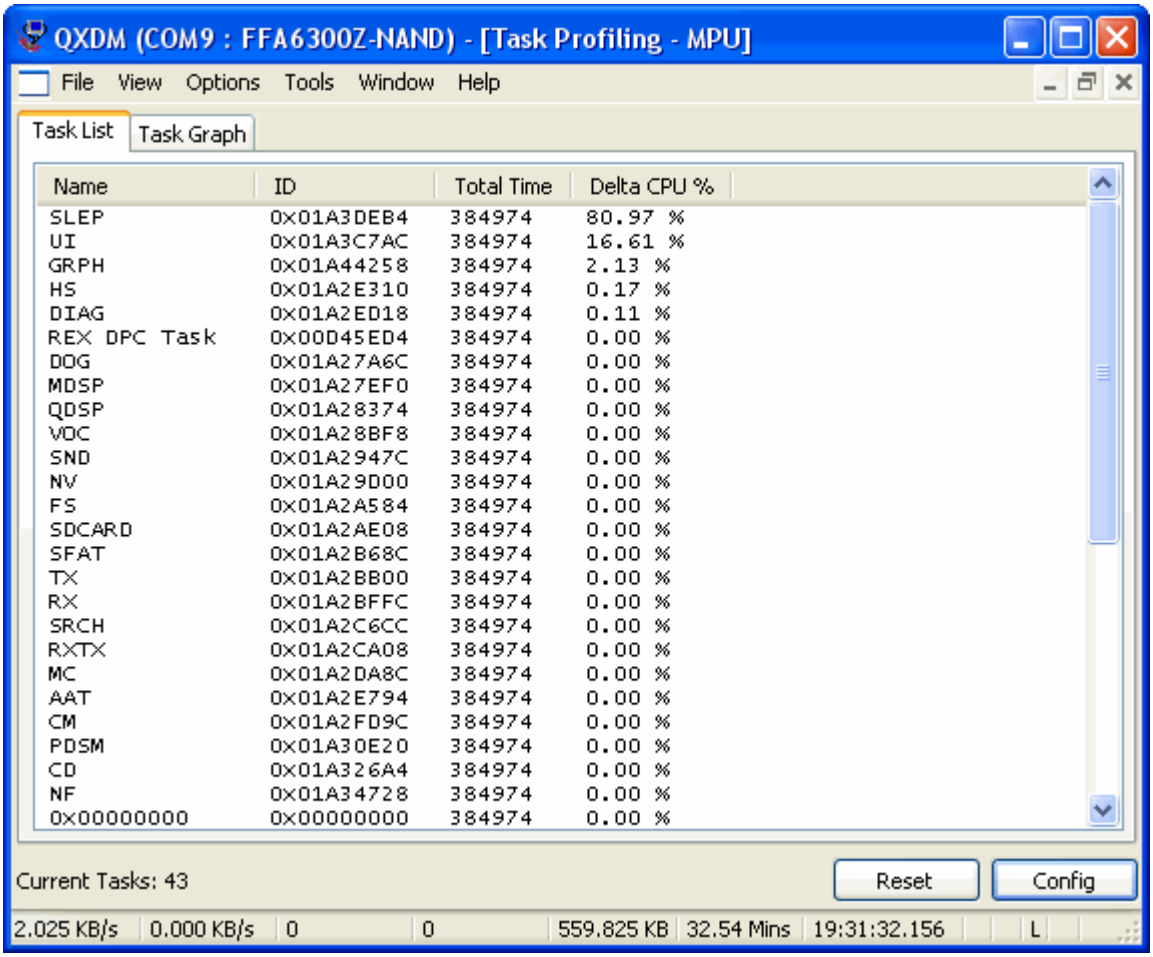

Figure 4-23 Task List Pane

NOTE Two Task Profiling views exist for dual-processor targets, Application Processing Unit (APU) and Modem Processing Unit (MPU).

The CONFIG button allows you to configure which columns to display in the task list, which tasks to graph, and the interval at which task information is sent.

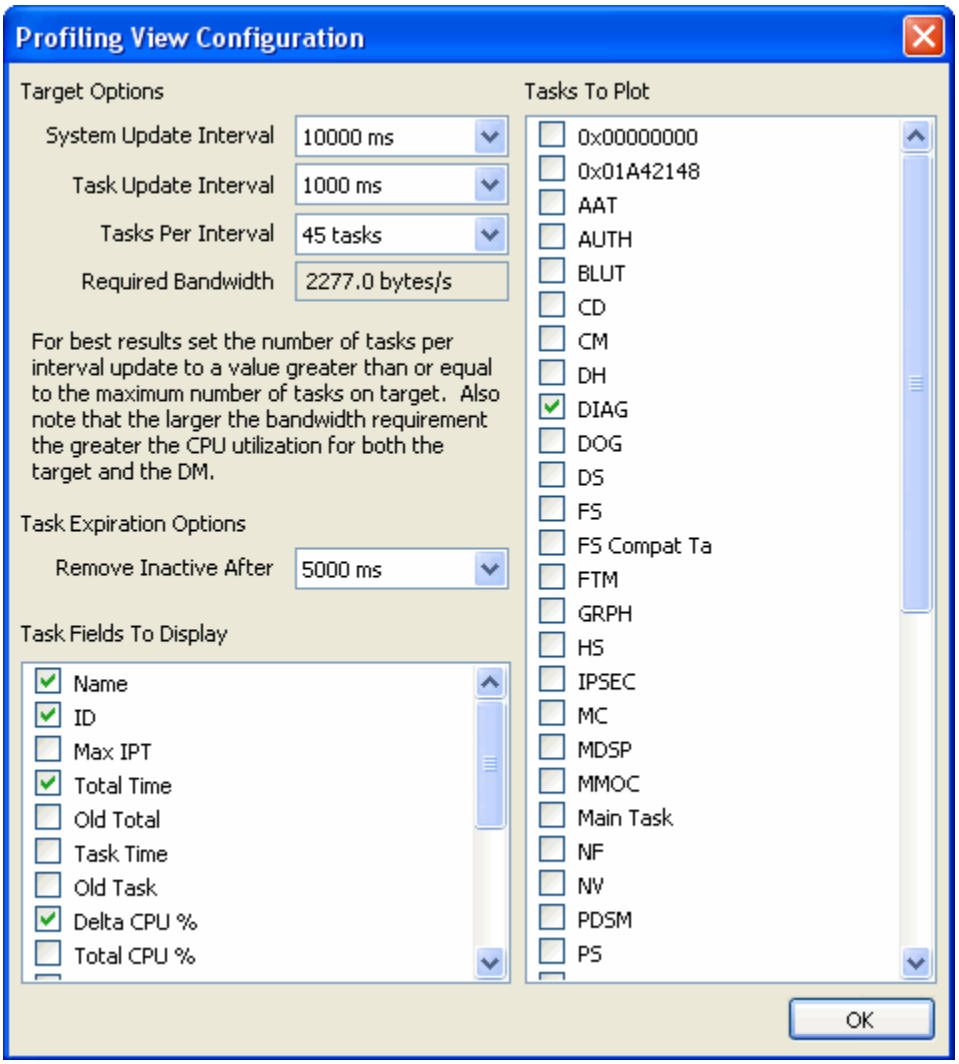

Figure 4-24 Task Profiling Configuration

The columns that can be displayed are described in the Task Profiling Task Fields table (Table 4-5). For additional information, refer to [Q1] for System-Level Task Profiling and Per-Task Profiling logs.

Table 4-5 Task Profiling Task Fields

Task	Description
Delta CPU %	Calculation is (Total Time – Old Total/Task Time – Old Task) converted to a percentage
ID	Unique identifier for the task
Locked	Interrupts locked state
Max IPT	Maximum interrupt time in ms that the task has kept interrupts locked
Name	Task name provided by the System Level Task Profiling log
Old Task	Previous update of Task Time
Old Total	Previous update of Total Time
Priority	The priority of the task
Signal Mask	Signal mask from the TCB that is set for the task
Suspended	Task suspended state
Stack Pointer	Stack pointer of the task
Stack Limit	Limit on the system stack for an ISR
Stack Size	Stack size in bytes of the task
Task Time	Total time task has been running since starting Task Profiling
Total CPU %	Percentage of time task has been running since starting Task Profiling
Total Time	Total time all tasks have been running since starting Task Profiling
Wait Mask	Signal mask of signals on which the task is currently waiting

4.5.8 Debug Trace View

Debug Trace View enables you to view the trace dump files that are written to EFS on the phone as a result of abnormal target failure. The contents of the trace dump are retrieved when a trace dump file is selected and START TRANSFER is clicked.

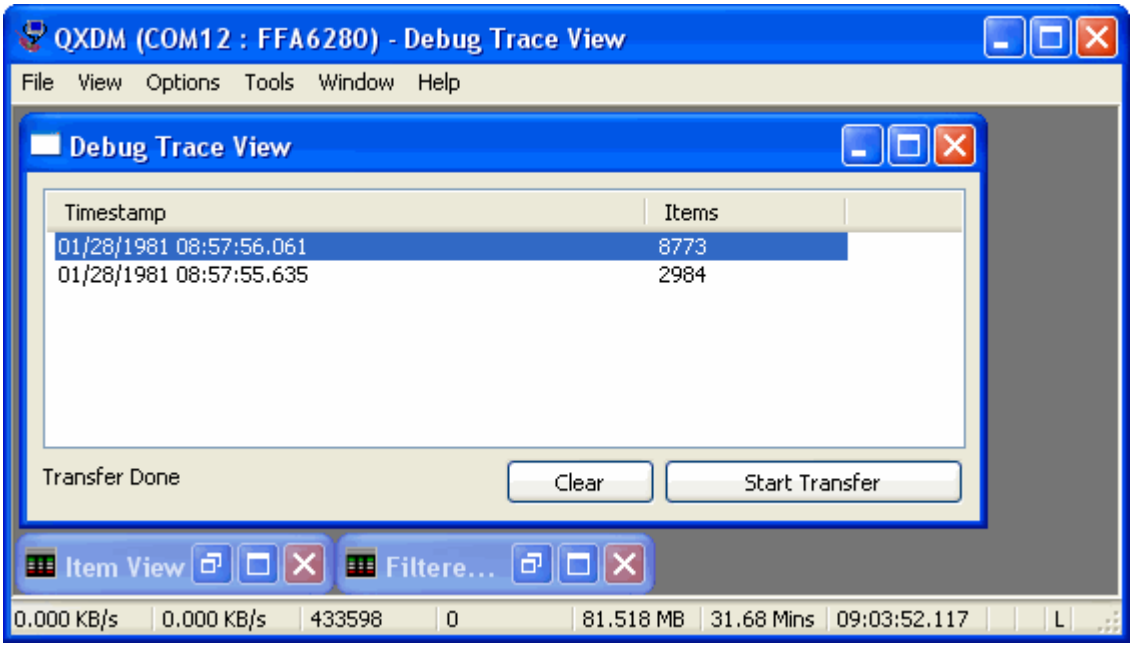

Figure 4-25 Debug Trace View dump files

NOTE To facilitate analysis of the Debug Trace dump file, it is recommended that a new ISF file be started before each transfer and then saved when the transfer has completed. Additionally, all views except the Debug Trace View and Item View should be closed to ensure that the ISF file contains the traffic stored in the retrieved Debug Trace dump file and the traffic necessary to retrieve the contents of the Debug Trace file.

The results can be viewed in the Item View or a Filtered View that has been configured to receive Debug/Get Trace Item Response items.

Figure 4-26 Debug Trace Dump retrieved items

4.5.9 CDMA Finger Placement

The subplots in this view are described below.

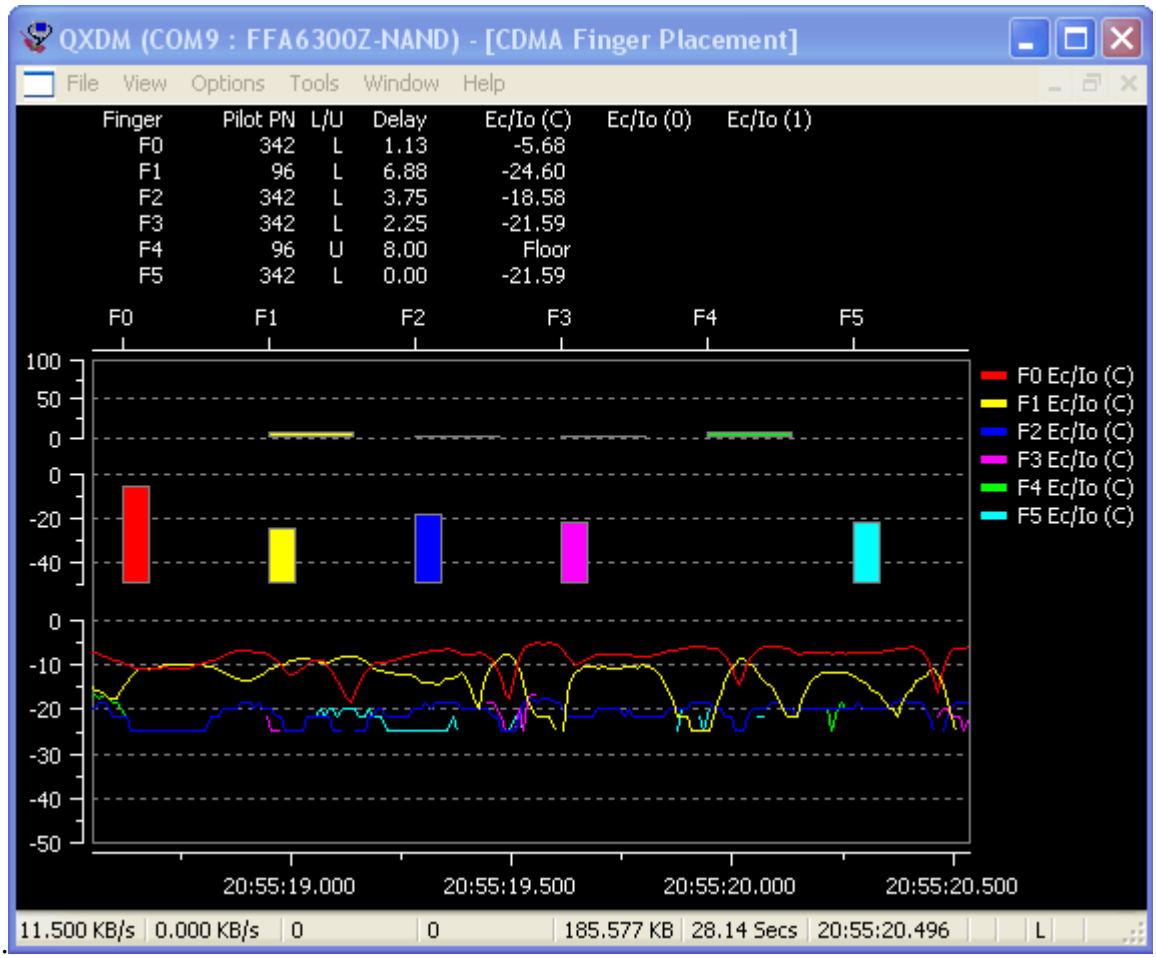

Figure 4-27 CDMA Finger Placement

4.5.9.1 Finger statistics

Finger statistics are displayed in this table.

4.5.9.2 Delay (chips)

This is the pilot PN of the finger, which the MSTR is tracking. The earliest arriving finger (smallest absolute position) is assumed to be the MSTR which is typically acceptable as the MSTR is currently not logged for CDMA.

4.5.9.3 Current E_c/I_0 (dB)

This is the instantaneous E_c/I_0 for each finger. For targets that support two antennae, three bars are displayed, Combined, Antenna 1, and Antenna 2.

4.5.9.4 E_c/I_0 (dB)

This is E_c/I_0 for each finger, based on time of arrival. For targets that support two antennae, lines are drawn for Combined, Antenna 1, and Antenna 2 for each finger.

NOTE The x-axis is time reported by the target, not generic time. This means that QXDM will ignore data from logs whose timestamp precedes the last timestamp plotted.

4.5.10 Status

The <F9> Status and Dynamic Status views display state information about the phone, such as Hardware Version, Build Version, System Determination, and Call Manager states.

5 Log View

5.1 Log View

The Log View is provided for backwards compatibility. It is a Filtered View with limited functionality that is configured and behaves according to the legacy requirements described below.

NOTE The legacy Log View has limited functionality and is provided only for backward compatibility. The correct approach is to use Always-On logging behavior in conjunction with Filtered Views.

Refer to Section 3.1.10 for details on how to configure Always-On logging behavior.

5.2 Log View Configuration

All client and view registrations, including the selections made through Options → Log View Configuration, contribute to what ends up in the log file. Log View behavior is as follows:

1. The Log View is created when the accelerator key, ALT + L, is pressed and is destroyed when ALT + L is pressed a second time.
2. The first time ALT + L is pressed, the current Item Store is cleared.
3. The second time ALT + L is pressed the current Item Store is saved to an ISF file with an automatically generated name consistent with the legacy DLF naming convention (MYYYYMMDD_HHMMSS.isf).
4. The contents of the ISF will be all items received between steps 2 and 3, not just the items displayed in the Log View list.
5. A new ISF log file is created on each phone (re)connection if Restart Log View Upon Disconnect from Options → Log View Configuration → Misc is checked.

NOTE The Log View Configuration is provided for legacy reasons and does not utilize the advanced features of QXDM Always-On logging (see Section 2.5.1). The behavior described above is equivalent to configuring a generic Filtered View, using the File → New Items (ALT + I) to restart logging, and using the File → Save Items (CTRL + I) to save the log session.

The following sections describe how to use the legacy Log View Configuration dialog.

5.2.1 Log Packets

The number of logs displayed in the Log Mask dialog will vary depending on the number of logs supported by the connected FFA or SURF.

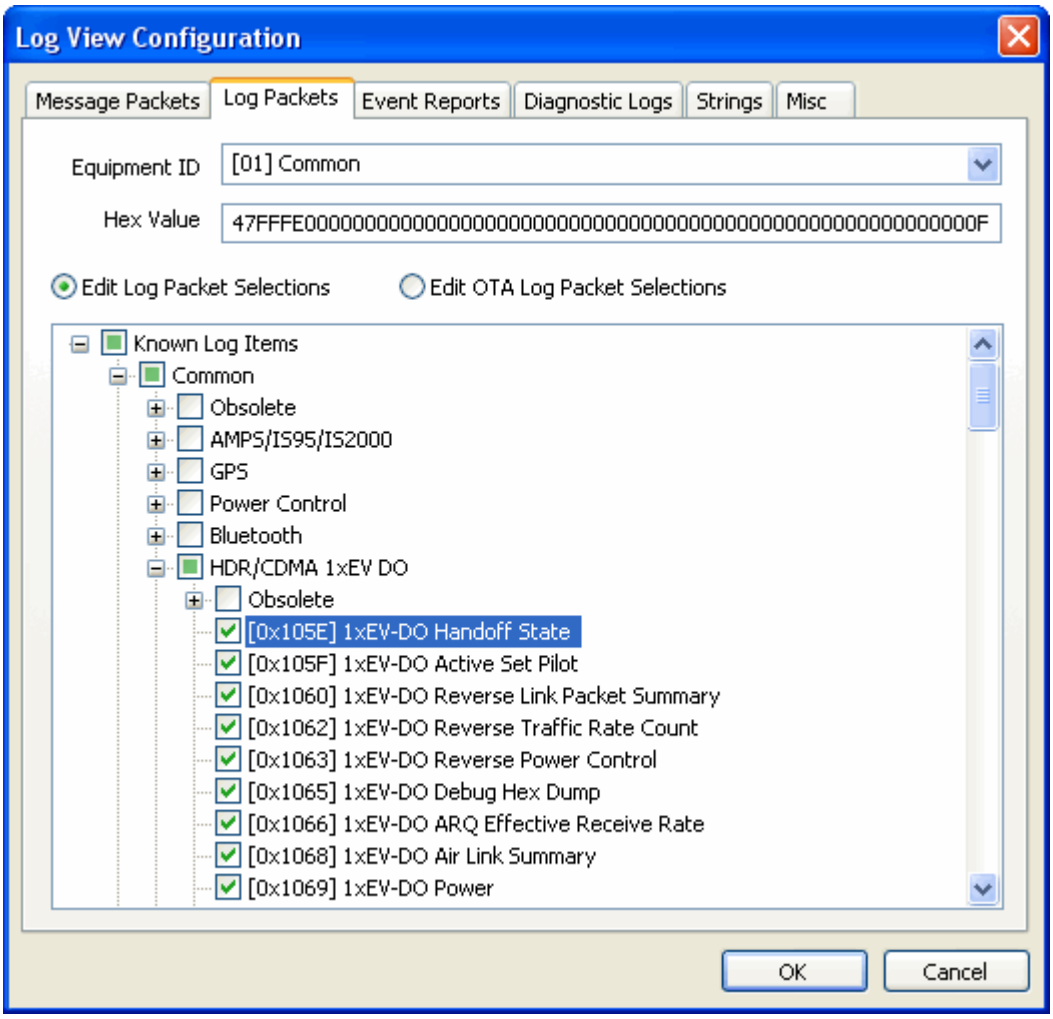

Figure 5-1 Options → Log View Configuration → Log Packets

5.2.2 Message Packets

The Message Packets tab allows you to configure the types of Debug Trace messages that are displayed by the Log View.

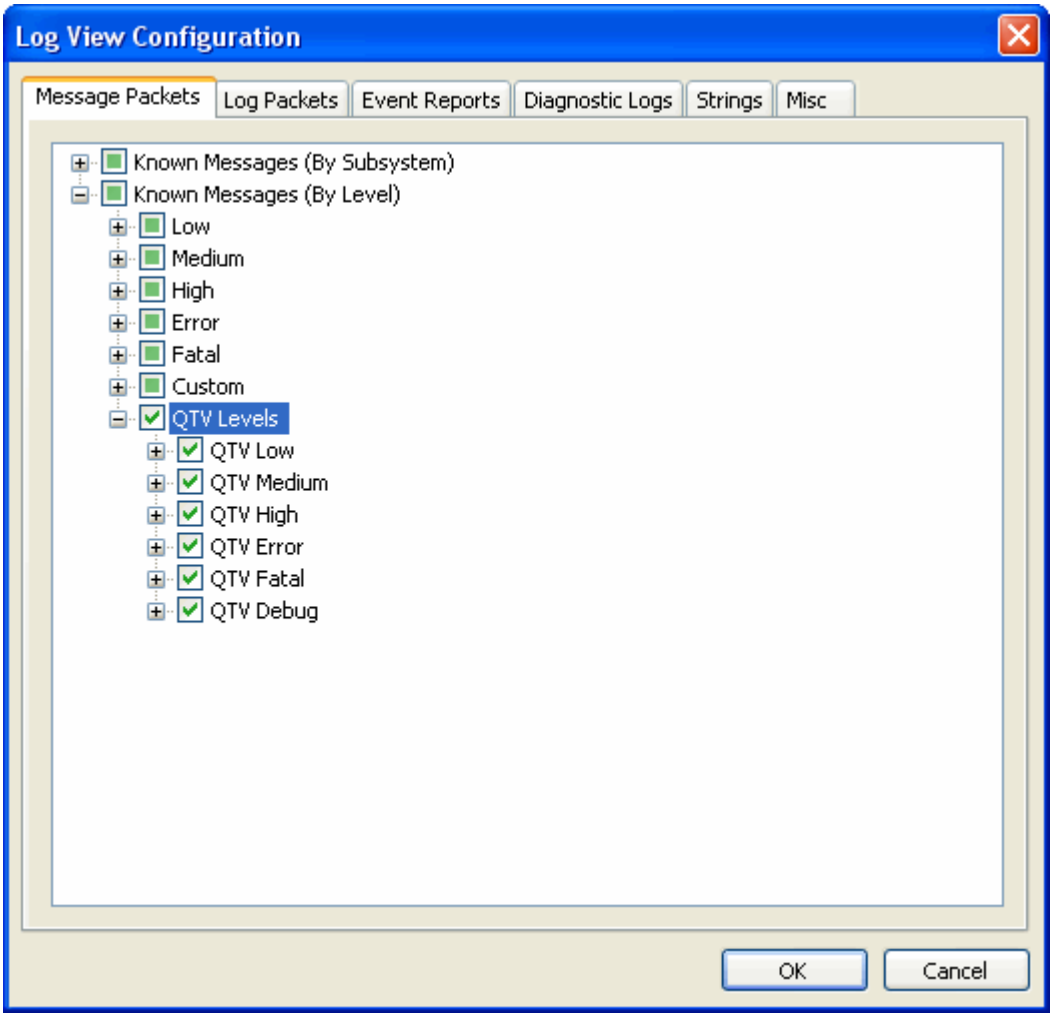

Figure 5-2 Options → Log View Configuration → Message Packets

5.2.3 Event Reports

The Event Reports tab allows you to control the events that are displayed by the Log View.

NOTE All events should be logged for maximum compatibility with some postprocessing tools.

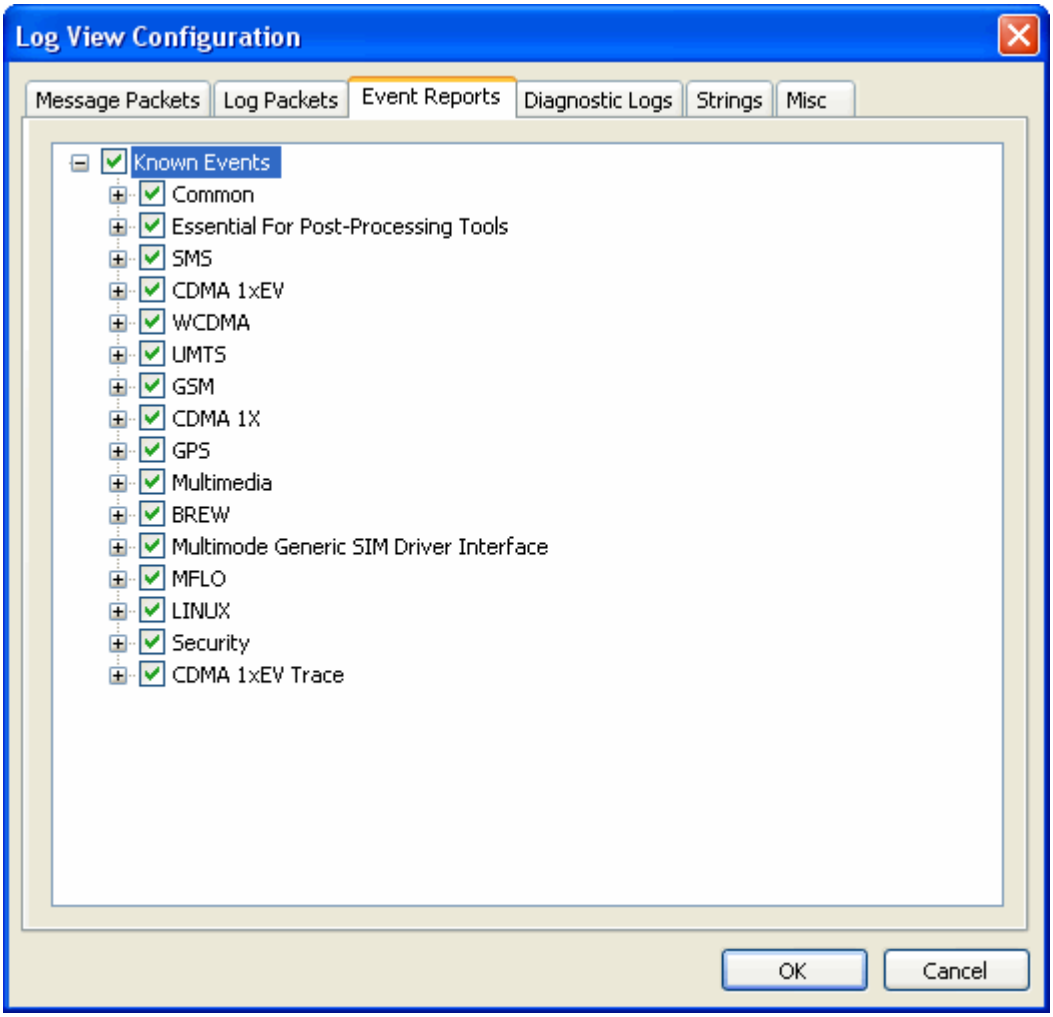

Figure 5-3 Options → Log View Configuration → Event Reports

5.2.4 Diagnostic Logs

The Diagnostic Logs tab allows you to specify the diagnostic log items that are generated by the Log View.

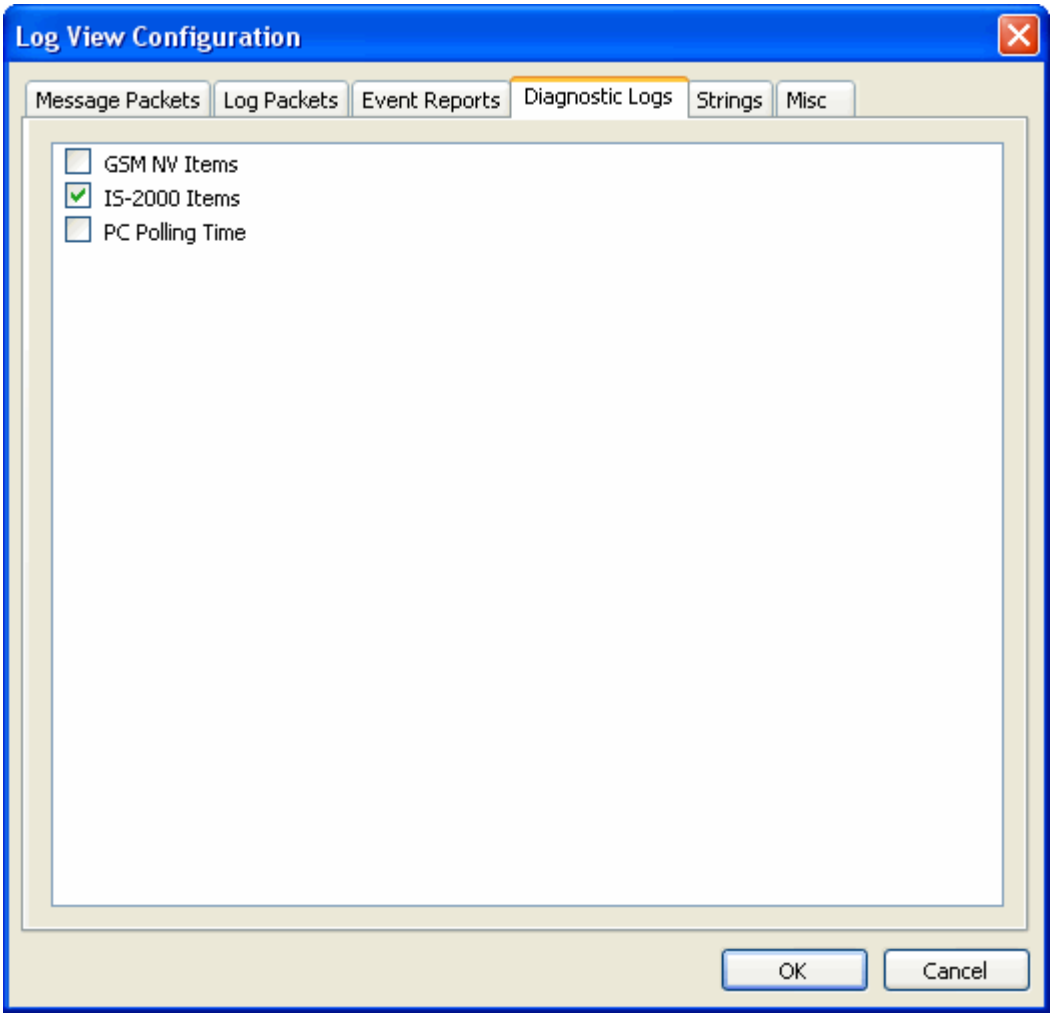

Figure 5-4 Options → Log View Configuration → Diagnostic Logs

5.2.4.1 GSM NV Items

The following GSM NV Items are requested and logged when this is checked:

- GSM NV Test Code Version
- GSM NV System Software Version
- GSM NV RF Cal Version
- GSM NV RF Cal Date
- GSM NV RF Load Date
- GSM NV RF Cal File
- GSM NV RF Config Version

5.2.4.2 IS-2000 Items

The following diagnostic packets are requested and logged every 500 ms when this is checked:

- IS 2000 Status (periodic)
- IS 2000 Standard Retrieve Parameters (periodic)
- IS 2000 Extended Retrieve Parameters (periodic)

5.2.4.3 PC Polling Time

A PC Polling Time record is logged on establishing connection when this is checked. This record is for internal use only by the QCAT post-processing tool.

5.2.5 Strings

The Strings tab allows you to specify the string items that are displayed by the Log View.

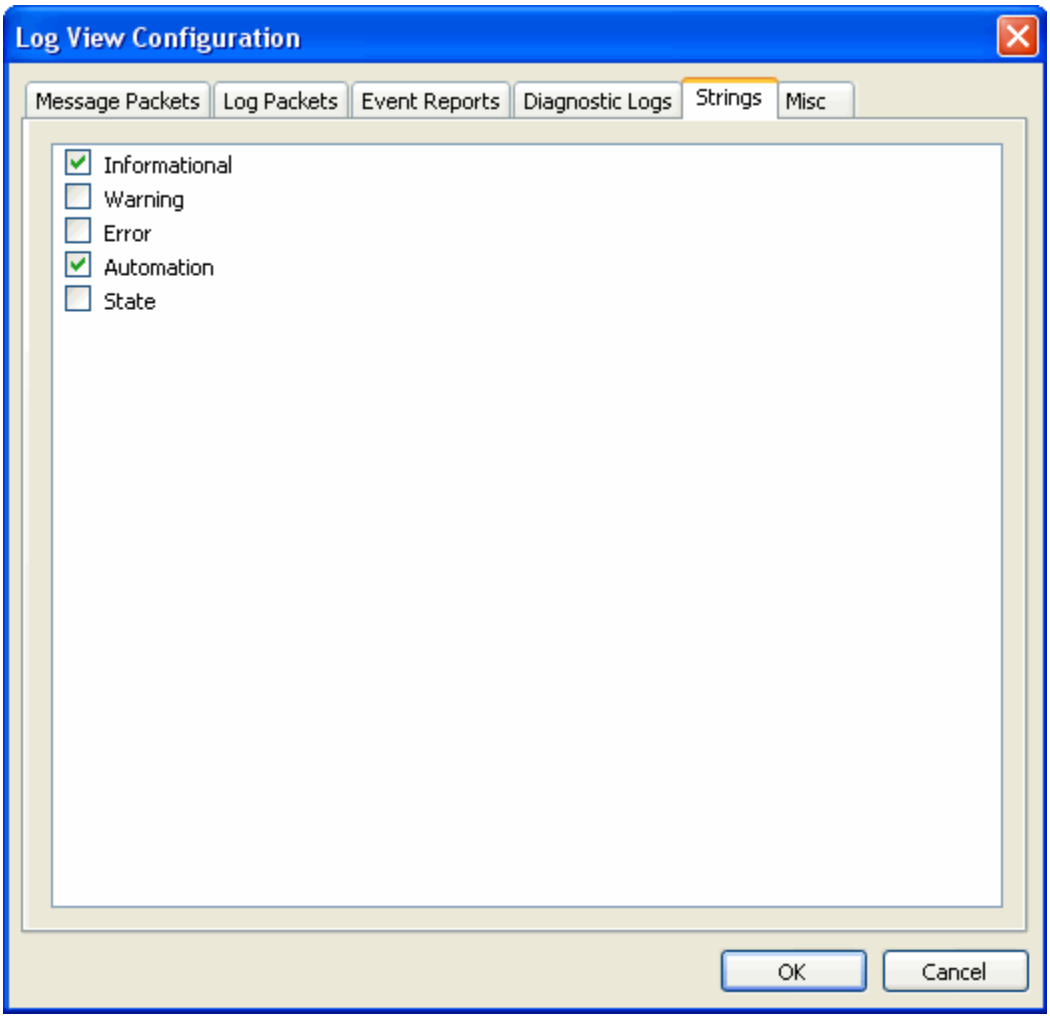

Figure 5-5 Options → Log View Configuration → Strings

5.2.6 Misc

The Misc tab allows you to specify where log files generated by the operation of the legacy Log View are saved. The default folder is the QXDM ISF folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\ISF). Checking Restart Log View Upon Disconnect results in a new log file each time QXDM is disconnected from a target while Log View is enabled.

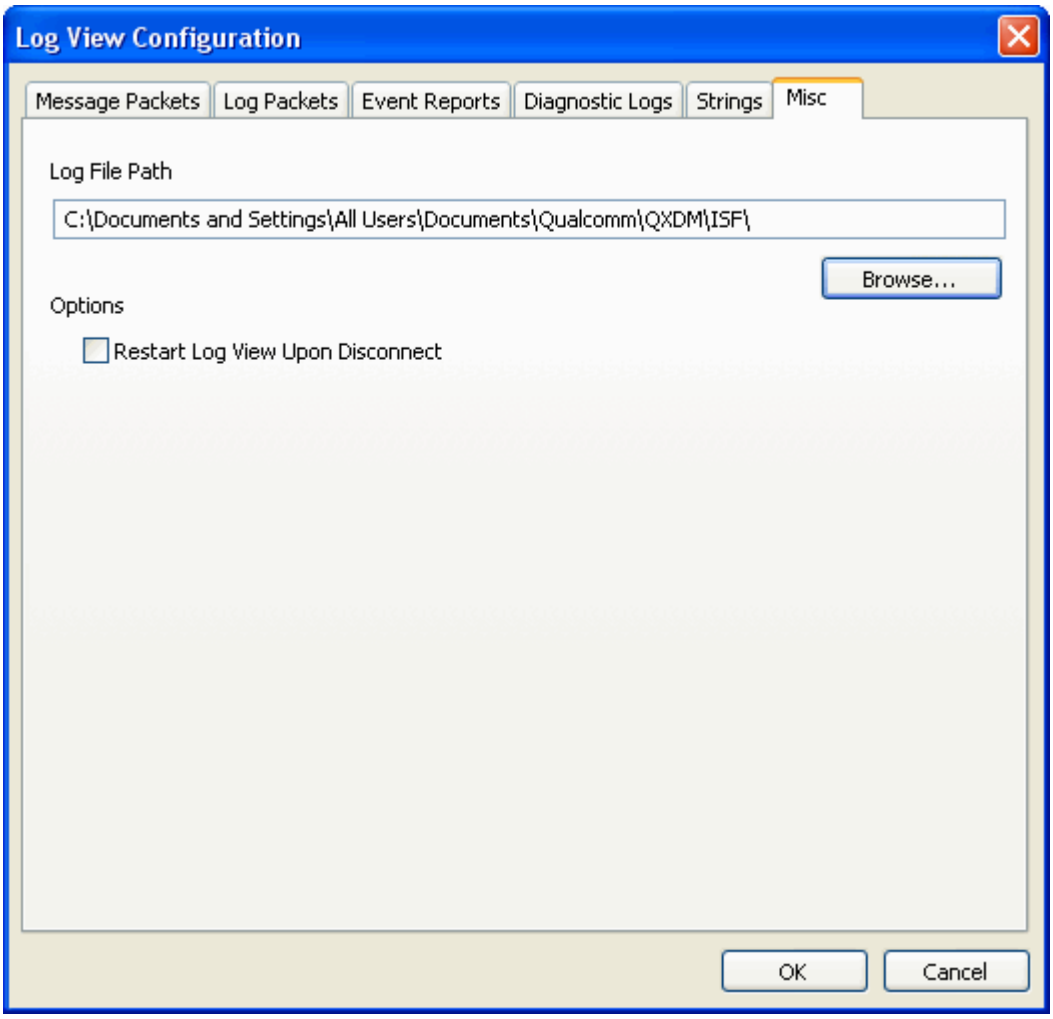

Figure 5-6 Options → Log View Configuration → Misc

6 Messages View

6.1 Messages View

Similar to Log View, the Messages View is a special case filtered view. For legacy reasons, the Messages View is still configured using the Message View Configuration dialog, accessible from the Options → Message View Configuration menu or by pressing CTRL+F5.

Messages View behavior is:

- Messages View shows only the traffic received during the life of the view.
- Message View Configuration settings are communicated to the target when the Messages View is created.
- Message View Configuration settings are removed from the target when the view is destroyed.

NOTE The Messages View has limited functionality and is provided only for backward compatibility. The correct approach is to use Filtered Views and configuration (.DMC) files.

6.2 Message View Configuration

The configuration dialog consists of five tabs, Message Packets, Log Packets, Log Packets (OTA), Event Reports, and Strings.

6.2.1 Log Packets

The Log Packets tab is shown in Figure 6-1.

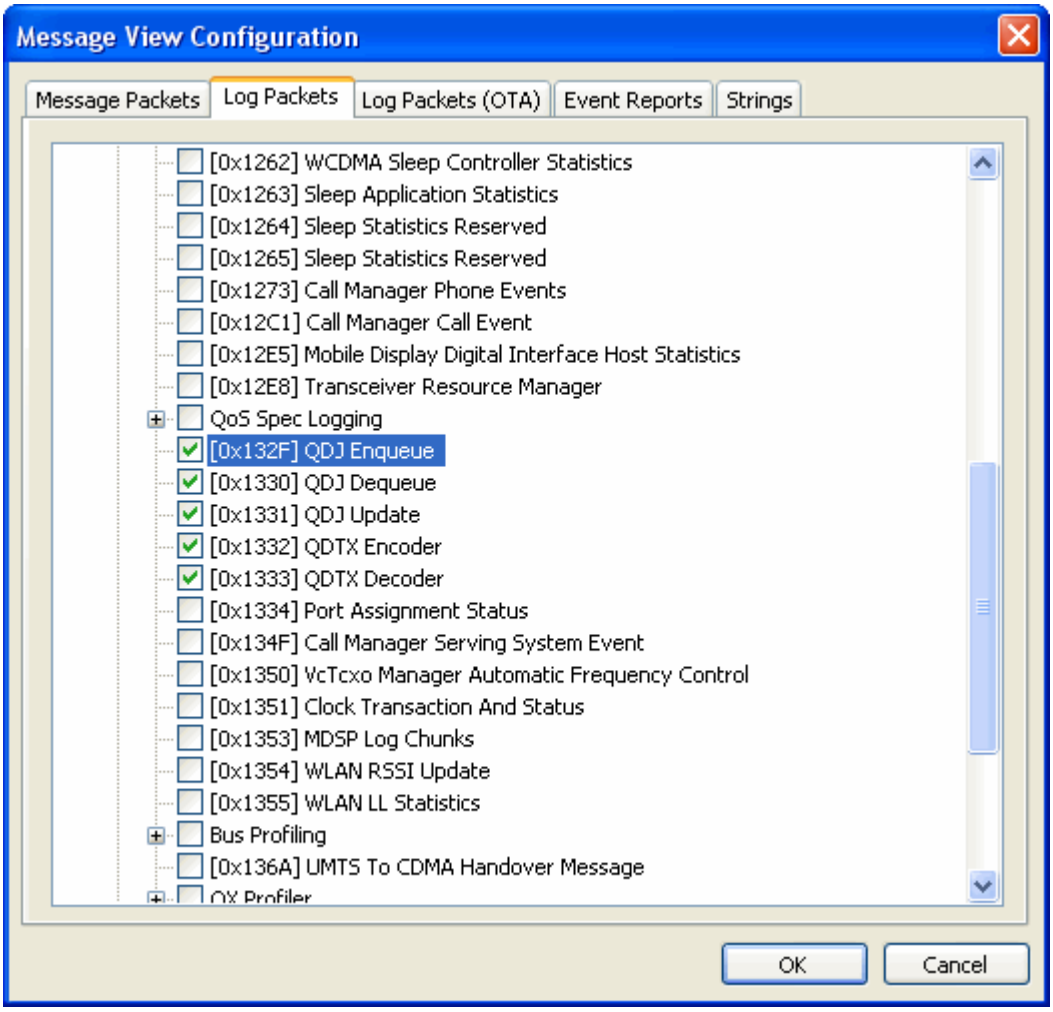

Figure 6-1 Options → Message View Configuration → Log Packets

6.2.2 Log Packets (OTA)

The Log Packets (OTA) tab allows you to select the OTA Log message types.

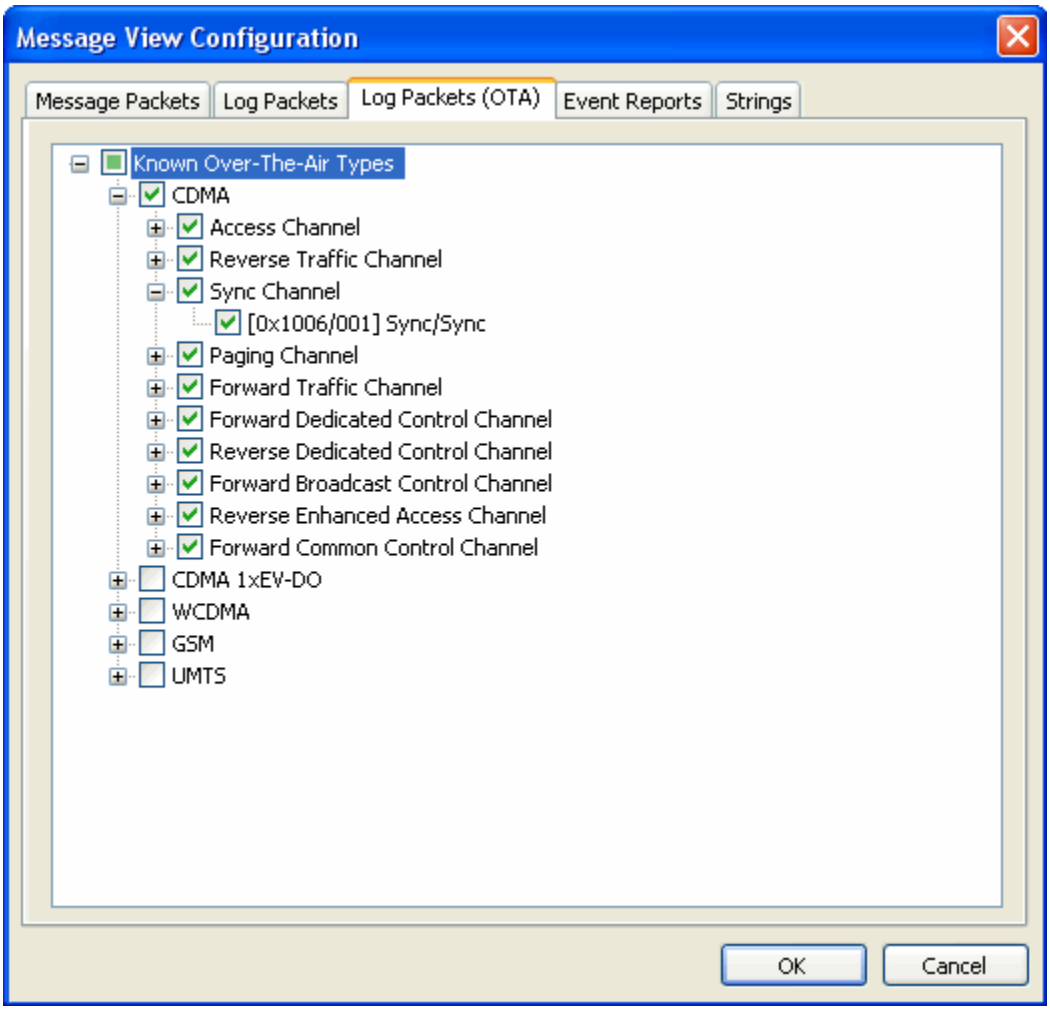

Figure 6-2 Options → Message View Configuration → Log Packets (OTA)

6.2.3 Message Packets

The Message Packets tab allows you to set the debug trace message types that the Messages View will accept.

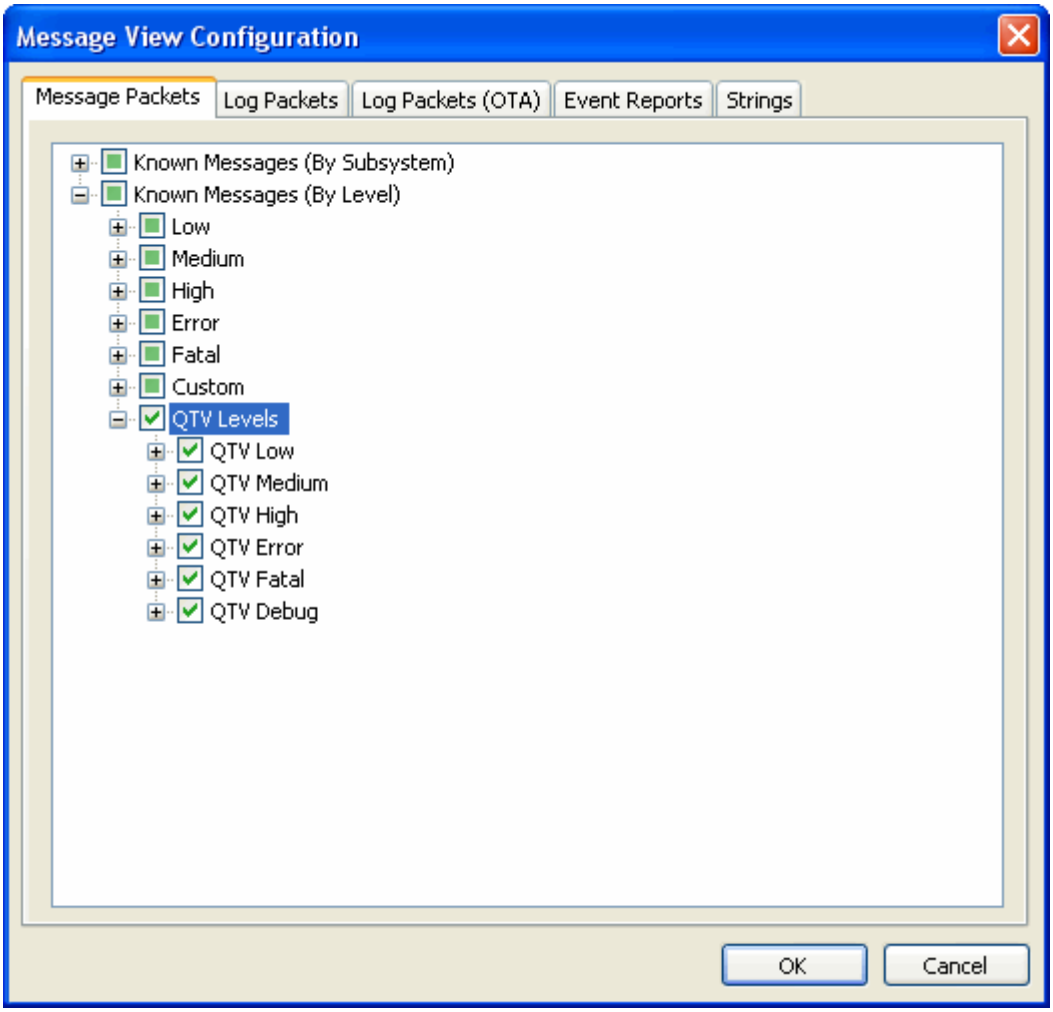

Figure 6-3 Options → Message View Configuration → Message Packets

6.2.4 Event Reports

The Event Reports tab allows you to control the events that the Messages View will accept.

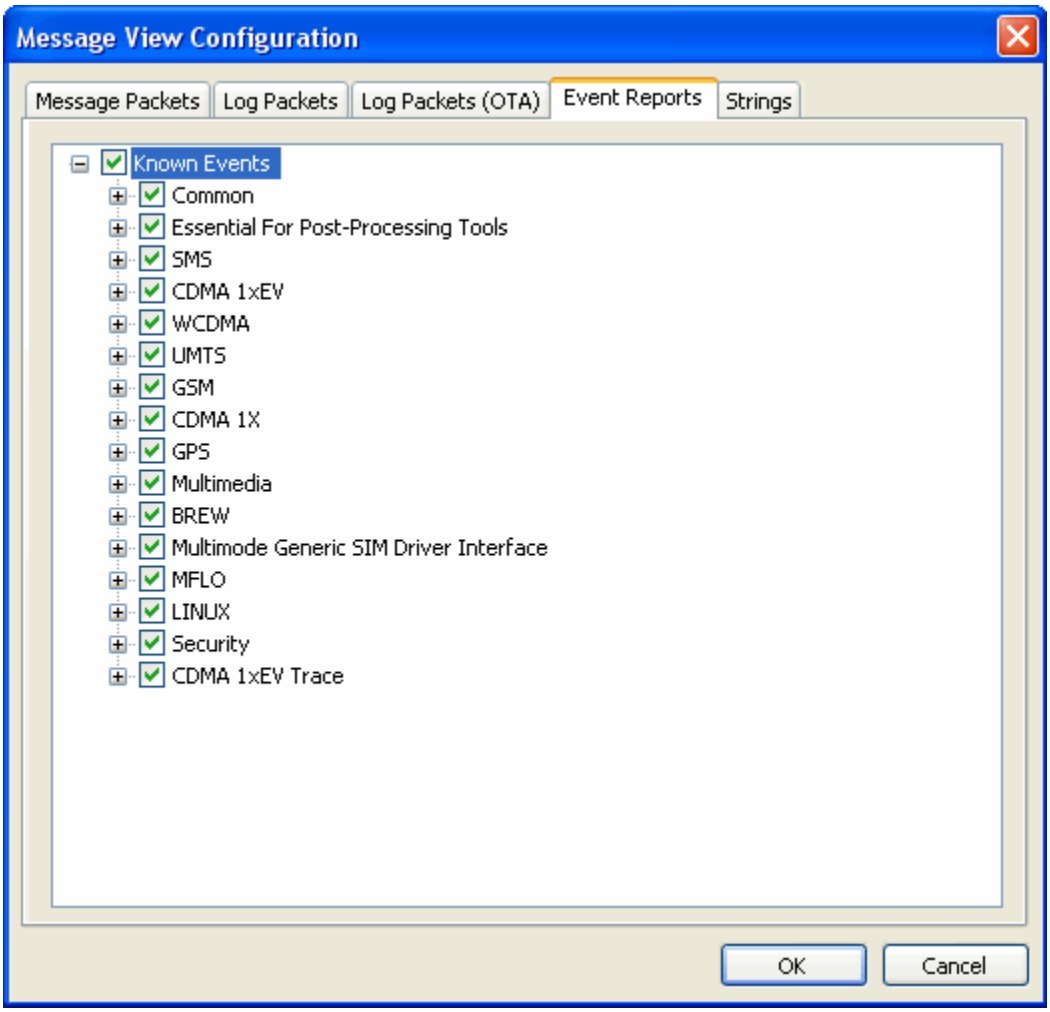

Figure 6-4 Options → Message View Configuration → Event Reports

6.2.5 Strings

The Strings tab allows you to control the types of string items that the Messages View will accept.

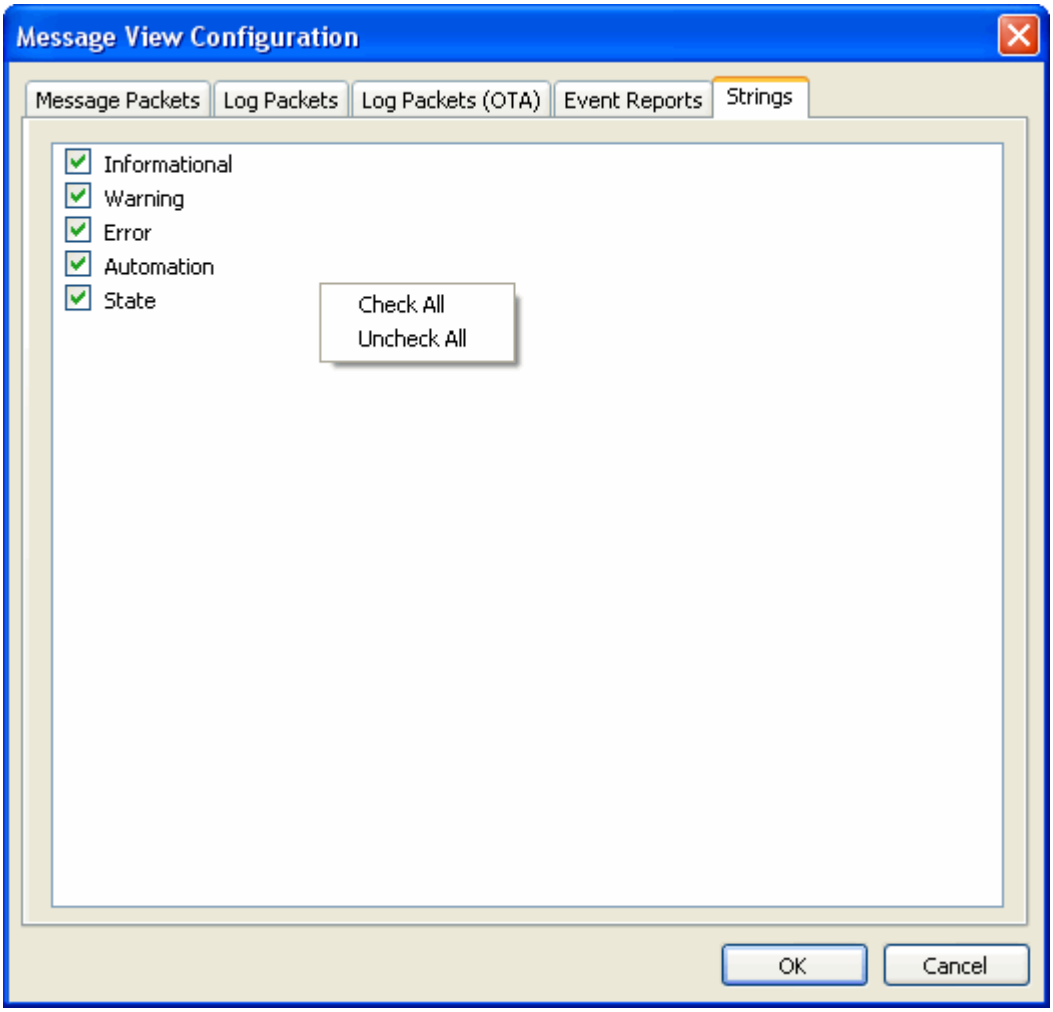

Figure 6-5 Options → Message View Configuration → String

7 Command Prompt Interface

QXDM provides an interface that allows you to enter and send commands manually or have them sent from a prepared script file (.SCR extension).

The interface is limited in that command status and response data can only be evaluated visually. QXDM COM automation (Chapter 8) does not have this limitation and is the preferred solution to write fully automated test cases.

7.1 QXDM properties

NOTE QXDM property definitions are not sufficient to describe many diagnostic packets and have been migrated to the QXDM database. User-defined property definitions will also need to be migrated to user database definitions. Use the Database Editor (included with QXDM) to describe user-defined items (see [Q6] for details).

Table 7-1 lists the replacements that should be used instead of the obsolete property interface commands. The database command replacements are described in Section 7.2.

Table 7-1 Property command replacements

Property command	Database command replacement
get_property <property name> <optional field>	RequestItem <item name> <optional fields> RequestNVItemRead <item name> <optional fields>
put_property <property name> <fields>	RequestItem <item name> <fields> RequestNVItemWrite <item name> <fields>
print_property <property name>	Click on an item in the Item Tester Application (Figure 7-1)
list_properties	Use the Item Tester Application (accessible from the Tools menu, Figure 3-18)
nv_read <NV property item>	RequestNVItemRead <item name> <optional fields>
nv_write <NV property item>	RequestNVItemWrite <item name> <fields>

7.2 Command functions

NOTE The Script Help page documents all supported command prompt functions. It is available in the View combo box from the View Bar (Figure 4-1). This view also provides documentation for the legacy script commands that are part of the original DOS DM scripting interface.

Some script commands direct packets to the test phone, while others affect only QXDM operations. Command output is displayed in the Command Output window and can also be viewed from the Item View and any Filtered View that has registered interest in automation string items.

Several functions are provided to support waiting for, requesting, and parsing of any item that is described in the QXDM database. Additional support is provided in the Item Tester application that provides formatting examples using these commands for all items described in the database. The Item Tester is accessible from the QXDM Tools menu.

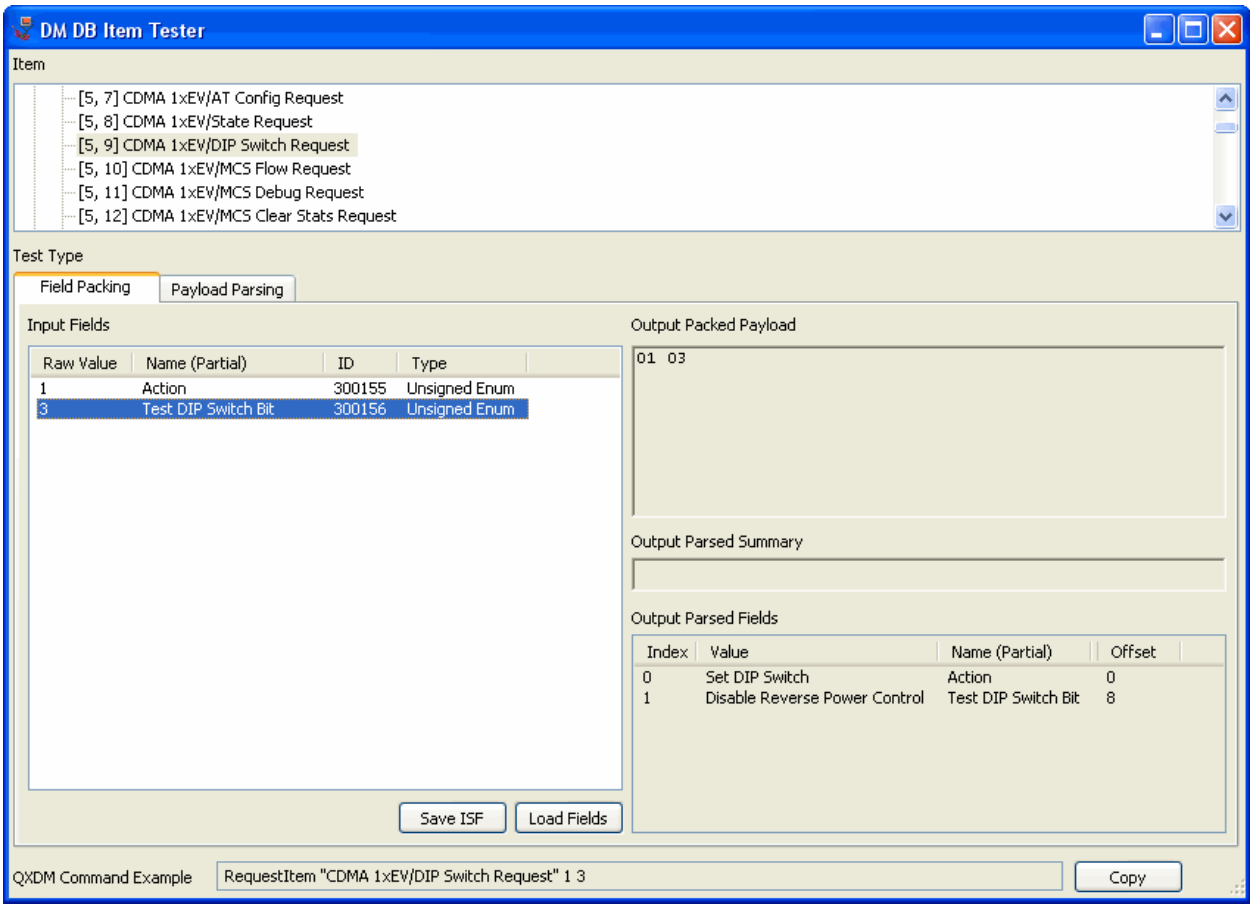

Figure 7-1 Item Tester QXDM Command Example

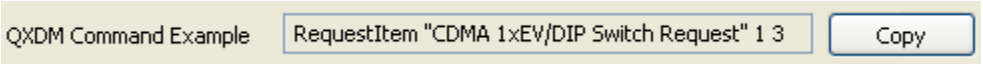

Figure 7-2 Copy from Item Tester application to QXDM command

NOTE Functions that wait for responses use the Script Commands timeout option configurable through the menu item Options → Communications → Script Commands.

7.2.1 CWait

This halts scripting for a specified amount of time.

Table 7-2 CWait parameters

Name	Description
Time	Time in centiseconds to wait

Example

```
Wait 500
Halt scripting for 5 sec
```

7.2.2 Echo

Table 7-3 Echo parameters

Name	Description
Text	Adds the provided string to the DM Item Store with the Automation type; the string will appear in a list-based view registered for this type (including the Command Output view)

Example

```
Echo "Test completed"
```

7.2.3 Logging

NOTE This command is provided only for legacy reasons and does not utilize the advanced logging features of QXDM Always-On logging (see Section 2.5.1).

This brings up or closes the Log View.

Table 7-4 Logging parameters

Name	Description
State	State can be either On or Off: ■ On – Brings up the Log View, if not already active ■ Off – Tears down the Log View, if active

Example

```
Logging On
```

7.2.4 LogMask

NOTE This command is provided only for legacy reasons and does not utilize the advanced logging features of QXDM Always-On logging (see Section 2.5.1).

LogMask sets the Common portion of the Log View log configuration mask. Log masks for other equipment are not supported using this command.

Table 7-5 LogMask parameters

Name	Description
Mask	Mask is the desired log mask in the format of Hex Value portion of the Log Config dialog

Examples

```
LogMask 99ff
```

7.2.5 Mode

This requests that the target perform a mode change.

Table 7-6 Mode parameters

Name	Description
Mode	Mode can be one of the following string values: ■ FTM – Change to Factory Test mode ■ LPM – Change to Low Power mode ■ Offline-A – Change to Offline Analog mode ■ Offline-D – Change to Offline Digital mode ■ Online – Change to Online Digital mode ■ Reset – Reset the target (may require Phone Offline mode)

Example

`Mode Reset` – Requests the target reset

7.2.6 Pause

This halts processing of a legacy script (.scr) processing. The Space Bar must be pressed in the Command edit box to continue processing.

Example

```
Pause
```

7.2.7 RequestItem

RequestItem allows you to schedule a request to be sent to the phone. The request itself is built according to a description entered into the QXDM database or the user database DIAG entity table. RequestItem waits for a response and, if received, parses it according to the default description in the QXDM database or the user database (if a description exists).

RequestItem parameters are described in Table 7-7.

Table 7-7 RequestItem parameters

Name	Description
Name	Name of DIAG entity to request (entity must be either a DIAG request or subsystem dispatch request)
Fields	List of space-separated fields used to build the request

Example

```
RequestItem "Version Number Request"
RequestItem "Extensible Parameter Retrieval Request" 1 1 100
```

7.2.8 RequestItemFile

RequestItemFile allows one to schedule a request to be sent to the phone using a file that contains input field values. The request itself is built according to a description entered into the QXDM database or a user database. RequestItemFile waits for a response and if received, parses it according to the default description in the QXDM database or a user database (if such a description exists).

Table 7-8 RequestItemFile parameters

Name	Description
Name	Name of DIAG entity to request (entity must be either a DIAG request or subsystem dispatch request)

FileName	Name of a file containing field values used to build the request. This argument can be omitted if the request does not require field values. The file should contain one line for each input field of field, value pairs, separated by the caret (^) character. For example:
	`Number Of Digits^10`
	`Digits^2144951009`
	`Service Option^17`
	Note: QXDM searches for FileName based on the following search criteria:
	■ Search using the file as-is
	■ Search using the installation default QXDM data path (C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM)
	■ Search using the Alternate Script Path (Options → Settings)

Examples

```
RequestItemFile "Mode Change Request" "ModeOnlineDigital.txt"
RequestItemFile "Call Origination Request" "CallOriginate.txt"
```

7.2.9 RequestNamedItem

RequestNamedItem allows you to schedule a request to be sent to the phone. It is similar to RequestItem with the exception that the named response must also be specified for QXDM to use in parsing the response. RequestNamedItem waits for the response and, if received, parses it according to the description in the QXDM database or the user database DIAG entity tables. Table 7-9 lists the parameter format.

Table 7-9 RequestNamedItem parameters

Name	Description
ResponseName	Name of DIAG entity to use for parsing the response
RequestName	Name of DIAG entity to request (entity must be either a DIAG request or subsystem dispatch request)
Fields	Field values used to build the request

Examples

```
RequestNamedItem "Version Number Response" "Version Number Request"
RequestNamedItem "My Response" "Extensible Parameter Retrieval Request" 1 1 125
```

7.2.10 RequestNamedItemFile

RequestNamedItemFile allows one to schedule a request to be sent to the phone using a file that contains input field values. The request itself is built according to a description entered into the QXDM database or a user database. RequestNamedItemFile waits for a response and if received, parses it according to the description matching the provided DIAG entity name.

Table 7-10 RequestNamedItemFile parameters

Name	Description
ResponseName	Name of the DIAG entity used to parse the responses (if received)
RequestName	Name of the DIAG entity to request (entity must be either a DIAG request or subsystem dispatch request)

FileName	Name of a file containing field values used to build the request. This argument can be omitted if the request does not require field values. The file should contain one line for each input field of field, value pairs, separated by the caret (^) character. For example:
	`Number Of Digits^10`
	`Digits^2144951009`
	`Service Option^17`
	Note: QXDM searches for "FileName" based on the following search criteria:
	■ Search using the file as-is
	■ Search using the QXDM installation default data path (C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM)
	■ Search using the Alternate Script Path (Options → Settings)

Examples

- RequestItemFile Mode Change Response Mode Change Request OnlineD.txt

  Builds and sends a Mode Change Request with field values contained in ModeOnlineDigital.txt to the target and waits for and parses the response using the database description for Mode Change Response.

- RequestItemFile Call Origination Response Call Origination Request Call.txt

  Builds and sends a Call Origination Request to the target with field values contained in the file CallOriginate.txt and waits for and parses the response using the database description for Call Origination Response.

7.2.11 RequestNVItemRead

RequestNVItemRead allows you to schedule one NV read request to be sent to the phone. The request itself is built according to a description entered into the QXDM database or the user database DIAG entity table. RequestNVItemRead waits for the response and, if received, parses it according to the description in the QXDM database or the user database DIAG entity tables. Table 7-11 lists the parameter format.

Table 7-11 RequestNVItemRead parameters

Name	Description
NVItemName	Name of the NV item entity to read
Fields	Field values used to build the request

Example

```
RequestNVItemRead "banner"
RequestNVItemRead "air_timer" 1
```

7.2.12 RequestNVItemWrite

RequestNVItemWrite allows you to schedule one NV write request to be sent to the phone. The request itself is built according to a description entered into the QXDM database or the user database DIAG entity table. RequestNVItemWrite waits for the response and, if received, parses it according to the description in the QXDM database or the user database DIAG entity tables. Table 7-12 lists the parameter format.

Table 7-12 RequestNVItemWrite parameters

Name	Description
NVItemName	Name of the NV item entity to write
Fields	Field values used to build the request

Example

```
RequestNVItemWrite "banner" "My MSM7500"
```

7.2.13 RequestNVItemIDRead

RequestNVItemIDRead allows one to schedule one NV read request to be sent to the phone. The request itself is built according to arguments entered as raw bytes. RequestNVItemIDRead waits for the response, reporting result status.

Table 7-13 RequestNVItemIDRead parameters

Name	Description
ItemNumber	Item number of the NV DIAG entity to be read
RawBytes	Optional list of space-separated byte values used to build the NV item being read

Example

```
RequestNVItemIDRead 71
RequestNVItemIDRead 10 0 0 0
```

7.2.14 RequestNVItemIDWrite

RequestNVItemIDWrite allows one to schedule one NV write request to be sent to the phone. The request itself is built according to arguments entered as raw bytes. RequestNVItemIDWrite waits for the response, reporting result status.

Table 7-14 RequestNVItemIDWrite parameters

Name	Description
ItemNumber	Item number of the NV DIAG entity to be written
RawBytes	Optional list of space-separated byte values used to build the NV item being written

Example

```
RequestNVItemIDWrite 71 77 83 77 32 54 53 53 48
RequestNVItemIDWrite 10 0 0 0
```

7.2.15 Run

This transfers command processing to a legacy (.scr) script file.

Table 7-15 Run parameters

Name	Description
FileName	FileName is the name of the legacy *.scr file to process. **Note:** QXDM searches for FileName based on the following search criteria: ■ Search using the file as-is ■ Search using the QXDM installation default data path (C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM) ■ Search using the Alternate Script Path (Options → Settings)

Example

```
Run "Test Case 20.scr"
Run "C:\My .SCR tests\Test Case 20.scr"
```

7.2.16 SendRawRequest

SendRawRequest allows one to schedule one DIAG request to be sent to the phone. The request itself is built according to arguments entered as raw bytes. SendRawRequest waits for the response, reporting result status.

Table 7-16 SendRawRequest parameters

Name	Description
RawBytes	List of space-separated byte values used to build the DIAG request being sent

Example

```
SendRawRequest 39 71 0 77 83 77 32 54 53 53 48
SendRawRequest 0
```

7.2.17 Wait

This halts scripting for a specified amount of time.

Table 7-17 Wait

Name	Description
Time	Time in seconds to wait

Example

```
Wait 10
```

7.2.18 WaitForItem

WaitForItem allows you to register interest in and wait for an item. The item name is the DIAG entity name in the QXDM database or the user database DIAG entity table. Requests for asynchronous commands (logs, events, and debug messages) may result in a registration being sent to the phone. WaitForItem can also be used to wait for synchronous requests or responses. At the conclusion of the command, the registration is revoked.

NOTE WaitForItem only waits for a response. It does not make a request. Use RequestItem (see Section 7.2.1) or RequestNamedItem (see Section 7.2.9) to request an item and wait for the response.

Table 7-18 lists the parameter format.

Table 7-18 WaitForItem parameters

Name	Description
ItemName	Name of DIAG entity to request

Example

```
WaitForItem EVENT_CALL_CONTROL_TERMINATED
WaitForItem "Searcher and Finger"
WaitForItem "Get IS-2000 Status Response"
```

8 COM Automation Interfaces

8.1 Overview

The QXDM application is a Windows 2000/Windows XP (and later) application that utilizes the existing DIAG serial interface via the Atlas/QPST phone server in order to establish a connection to a phone target. Once a connection is established, QXDM allows the user to manipulate and monitor the phone target by ultimately sending and receiving DIAG traffic to and from the target. QXDM itself is built upon the DM Core Framework, a library of objects and methods that encompasses all nonuser interface-related functionality implemented by QXDM. This includes but is not limited to target connection management, target data partitioning, target data representation, and data storage and nonsequential data access.

To support extensibility and rapid prototyping, QXDM itself exports a subset of the DM Core Framework functionality through a set of COM interfaces. These interfaces allow QXDM to support scripting through any COM-compliant scripting language, e.g., VBScript, JScript, and Perl. Additionally, a large subset of these interfaces is exposed through HTML-based QXDM views. Each view combines the capabilities of the Microsoft Internet Explorer control, hosted in a QXDM window, with the rich QXDM COM interface capabilities, allowing both the QXDM team itself and QXDM users to create custom views.

The QXDM COM interfaces are roughly divided into five modules, each described in the following sections:

- QXDM – The QXDM COM automation interface that existed and shipped in QXDM versions prior to 03.05.50; although these interfaces are still part of the QXDM code base, it is intended that support for some of the interfaces will be phased out over time
- Client – Provides a means to partition QXDM items (data obtained from the target and internal QXDM processing) into unique collections; also provides direct access to QXDM items
- Client Config – Provides a means to configure the particular types of items in which a QXDM interface client is interested; i.e., it defines the partition a client represents
- Item – Interface to a particular item being managed by QXDM
- Field – Provides access to the QXDM database parsed fields of a particular item

NOTE Numerous automation sample scripts (written in both Perl and JScript) can be found under the Automation Samples folder.

8.2 QXDM Interface (IQXDM)

This section describes the legacy QXDM COM interface (IQXDM).

8.2.1 AppVersion

The AppVersion property returns the version of QXDM.

A BSTR value is returned.

8.2.2 ClearViewItems

The ClearViewItems() function clears all items in the specified view. The function operates as shown in Table 8-1.

Table 8-1 ClearViewItems parameter

Name	Type	Description
Display Name	BSTR	Name of display to clear; the name is interpreted as: ■ Item view – Item Store view ■ Messages view – Messages view ■ All others – Tagged filtered view name

The function returns a VARIANT_BOOL response:

- FALSE – Request was not received (e.g., view was not found)
- TRUE – Request was successfully received

8.2.3 COMPort

The COMPort property can be used to view the currently selected COM port or to change to another COM port for which QXDM will attempt to connect to a phone.

A SHORT value is returned:

- -1 – Error occurred
- 0 – Disconnect state
- 1…n – COM port selected

8.2.4 CopyViewItems

The CopyViewItems() function copies all items to an Item Store Format file. The function operates as shown in Table 8-2.

Table 8-2 CopyViewItems parameters

Name	Type	Description
Display Name	BSTR	Name of display to copy; the name is interpreted as: ■ Item view – Item Store view ■ Messages view – Messages view ■ Log view – Log view ■ All others – Tagged filtered view name
Destination File	BSTR	Name of the destination Item Store Format file

The function returns a VARIANT_BOOL response:

- FALSE – Request was not received (e.g., view was not found)
- TRUE – Request was successfully received

8.2.5 CloseView

The CloseView() function allows one to close an existing QXDM view. The function operates as shown in Table 8-3.

Table 8-3 CloseView parameters

Name	Type	Description
pViewName	BSTR	Name of QXDM view to close
pViewTag	BSTR	Window tag for views that support multiple instances or empty string ("")

The function returns a VARIANT_BOOL that indicates whether a view was closed.

NOTE Due to the unique operation of the QXDM log view, this function does not apply to the QXDM log view.

8.2.6 CreateView

The CreateView() function allows one to bring up a QXDM view. The function operates as shown in Table 8-4.

Table 8-4 CreateView parameters

Name	Type	Description
pViewName	BSTR	Name of QXDM view to bring up
pViewTag	BSTR	Window tag for views that support multiple instances or empty string (“”)

The function returns a VARIANT_BOOL that indicates whether a view was created. If a view of the same name is already present and that view does not support multiple instances, the existing view will be brought to the forefront. In this case, success is returned.

NOTE Due to the unique operation of the QXDM log view, this function does not apply to the QXDM log view.

8.2.7 DipSwitchMask

The DipSwitchMask() property is used to set DIP switch settings. It operates as shown in Table 8-5.

Table 8-5 DipSwitchMask parameter

Name	Type	Description
DIP switch value	SHORT	DIP switch values to set

The function returns no value.

8.2.8 ExportViewText

The ExportViewText() function exports all items to a text file. The function operates as shown in Table 8-6.

Table 8-6 ExportViewText parameters

Name	Type	Description
Display Name	BSTR	Name of display to export; the name is interpreted as: ■ Item view – Item Store view ■ Messages view – Messages view ■ Log view – Log view ■ All others – Tagged filtered view name
Destination File	BSTR	Name of the destination text file

The function returns a VARIANT_BOOL response:

- FALSE – Request was not received (e.g., view was not found)
- TRUE – Request was successfully received

8.2.9 GetIQXDM2

The GetIQXDM2() function allows you to obtain an interface pointer to the QXDM client interface model (IQXDM2) as discussed in Section 8.3. The command takes no arguments and returns an IDispatch interface pointer.

8.2.10 GPSPort

The GPSPort property can be used to view the currently selected COM port or to change to another COM port for which QXDM will attempt to connect to a GPS receiver.

A SHORT value is returned:

- -1 – Error occurred
- 0 – Disconnect state
- 1…n – COM port selected

8.2.11 IsPhoneConnected

The IsPhoneConnected property returns 1 if QXDM is in the phone Connected state and 0, otherwise. Use GetServerState (Section 8.3.3) to determine what Nonconnected state QXDM is in.

A LONG value is returned.

8.2.12 LoadConfig

The LoadConfig() function loads a configuration file (.DMC). The function operates as shown in Table 8-7.

Table 8-7 LoadConfig parameter

Name	Type	Description
Config File	BSTR	Configuration file (see Section 3.1.4 for details)

The function returns no response.

8.2.13 LoadItemStore

The LoadItemStore() function allows you to load an Item Store Format (*.ISF) file into QXDM. The function operates as shown in Table 8-8.

Table 8-8 LoadItemStore parameters

Name	Type	Description
pFileName	BSTR	Fully qualified path to Item Store Format file

The function returns a VARIANT_BOOL that indicates whether the Item Store Format file was loaded.

NOTE An ISF can be loaded into QXDM only when QXDM is in the Disconnected state.

8.2.14 LogMask

The LogMask property is enabled for legacy reasons only. The LogMask property returns the log mask that is configured for equipment ID 1 logging using the Log View.

A BSTR value is returned in the format of the Log View Configuration dialog.

8.2.15 LogMaskOff

The LogMaskOff() function turns off a log item for which the Log View is registered. The function operates as shown in Table 8-9.

Table 8-9 LogMaskOff parameters

Name	Type	Description
Log	SHORT	Log ID or log code
Equipment ID	SHORT	Equipment ID

The function returns no response.

8.2.16 LogMaskOn

The LogMaskOn() function turns on a log item for which the Log View is registered. The function operates as shown in Table 8-10.

Table 8-10 LogMaskOn parameters

Name	Type	Description
Log ID	SHORT	Log ID or log code
Equipment ID	SHORT	Equipment ID

The function returns no response.

8.2.17 OfflineDigital

The OfflineDigital() function attempts to set the phone to the Offline-Digital state.

The function returns a LONG value:

- 0 – Failure to set the phone to Offline-Digital state
- 1 – Successfully set the phone to Offline-Digital state

8.2.18 ResetPhone

The ResetPhone() function resets the phone.

The function returns a LONG value:

- 0 – Failure to reset the phone
- 1 – Successfully reset the phone

8.2.19 SaveConfig

The SaveConfig() function saves a configuration file (.DMC). The function operates as shown in Table 8-11.

Table 8-11 SaveConfig parameters

Name	Type	Description
Config File	BSTR	Configuration file (see Section 3.1.5 for details)

The function returns no response.

8.2.20 SendDmIcdPacket

The SendDmIcdPacket() function is deprecated. Use SendDmIcdPacketEx instead (see Section 8.2.21).

8.2.21 SendDmIcdPacketEx

The SendDmIcdPacketEx() function sends a DM request/response packet to the target. The function operates as shown in Table 8-11.

Table 8-12 SendDmIcdPacketEx parameters

Name	Type	Description
Packet	VARIANT	Safe array of bytes to send to the target
Timeout	LONG	Timeout value for the request

The function returns a VARIANT that contains the response from the target.

8.2.22 SendScript

The SendScript() function sends a command to QXDM to be executed on the QXDM command line. The function operates as shown in Table 8-13.

Table 8-13 SendScript parameter

Name	Type	Description
Command	BSTR	Command string to execute

The function returns no value.

8.2.23 SetCDMAProtocolRevision

The SetCDMAProtocolRevision() function allows you to set the CDMA protocol revision that is used when parsing CDMA Over-the-Air (OTA) log items. The function operates as shown in Table 8-14.

Table 8-14 SetCDMAProtocolRevision parameters

Name	Type	Description
rev	ULONG	Desired CDMA protocol revision; possible values are: ■ 0 – IS-95A ■ 1 – TBD ■ 2 – TBD ■ 3 – TBD ■ 4 – IS-95B ■ 5 – TBD ■ 6 – IS-2000 Rev 0 ■ 7 – IS-2000 Rev A ■ 8 – IS-2000 Rev B ■ 9 – IS-2000 Rev C ■ 10 – IS-2000 Rev C ■ 11 – IS-2000 Rev D ■ 0x7FFFFFFF – Automatic

The function returns no value.

8.2.24 SetWCDMAProtocolRevision

The SetWCDMAProtocolRevision() function allows you to set the WCDMA protocol revision that is used when parsing WCDMA OTA log items. The function operates as shown in Table 8-15.

Table 8-15 SetWCDMAProtocolRevision parameters

Name	Type	Description
rev	ULONG	Desired WCDMA protocol revision; possible values are: ■ 0 – 1999 (RRC 25.331 V3.5.0 – 12/00 3GPP) ■ 1 – 6/01 (RRC 25.331 V3.7.0) ■ 2 – 3/02 (RRC 25.331 V3.A.0) ■ 3 – 3/04 (RRC 25.331 V3.I.0) ■ 4 – 6/04 (RRC 25.331 V5.9.0) ■ 5 – 12/05 (RRC 25.331 V6.8.0) ■ 6 – 3/06 (RRC 25.331 V6.9.0) ■ 7 – 6/06 (RRC 25.331 V6.A.0) ■ 8 – 9/06 (RRC 25.331 V6.B.0) ■ 0x7FFFFFFF – Automatic

The function returns no value.

8.2.25 SaveItemStore

The SaveItemStore() function allows you to save a temporary Item Store Format file (*.ISF). This function is subject to the same conditions as Save Items (see Figure 3-7).

Table 8-16 SaveItemStore parameter

Name	Type	Description
pFileName	BSTR	Fully qualified file name of the Item Store Format file to be saved

The function returns no value.

NOTE When QXDM is run, an ISF is created using a temporary name in the QXDM ISF folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\ISF). This is then used as the current ISF. If the user (either through the interaction with the QXDM GUI or through the QXDM automation interface) initiates the loading of an existing ISF, the current ISF may be saved if it is nonempty and the user has enabled Item Store saving. In this case, the filename specified by SaveItemStore() is used to rename and move the current ISF. This process does not apply when the user has already loaded an existing ISF.

8.2.26 SetLoggingOn

The SetLoggingOn() function is deprecated. Use SetLoggingOnEx instead (see Section 8.2.27).

8.2.27 SetLoggingOnEx

The SetLoggingOnEx() function sets the log mask to the mask provided for the specified equipment ID and switches logging on. The function operates as shown in Table 8-17.

Table 8-17 SetLoggingOnEx parameter

Name	Type	Description
Log Mask	BSTR	Hexadecimal log mask in the Log View Configuration dialog format
Equipment ID	SHORT	Equipment ID intended for log mask

The function returns no value.

8.2.28 SetLoggingOff

The SetLoggingOff() function sets the log filename to the name provided as a parameter and switches logging off. The function operates as shown in Table 8-18.

Table 8-18 SetLoggingOff parameter

Name	Type	Description
Log filename	BSTR	Filename to use for logging

The function returns no value.

8.2.29 SetVendorDatabase

The SetVendorDatabase() function allows you to load a new vendor-specific database. A vendor-specific database is used when parsing items specific to a particular handset vendor (currently only NV items). A vendor-specific database also contains mobile model names. The function operates as shown in Table 8-19.

Table 8-19 SetVendorDatabase parameters

Name	Type	Description
pDatabasePath	BSTR	Fully qualified folder name containing the vendor database

The function returns a VARIANT_BOOL indicating success/failure.

8.2.30 QuitApplication

The QuitApplication() function causes the QXDM application to terminate.

The function returns no value.

8.2.31 QXDMTextOut

The QXDMTextOut() function adds a string to the Item Store. The function operates as shown in Table 8-20.

Table 8-20 QXDMTextOut parameter

Name	Type	Description
Text	BSTR	String to display in views that are registered to receive Automation strings

The function returns no value.

8.2.32 Visible

The Visible property can be used to get the Visible state of QXDM or to show or hide the QXDM server application. It operates as shown in Table 8-21.

Table 8-21 Visible parameter

Name	Type	Description
Visible value	VARIANT_BOOL	QXDM server application Visible state

The function returns a VARIANT_BOOL.

8.2.33 WaitEvent

The WaitEvent() function waits for an event from the target. It returns the event if received within the timeout specified; otherwise, a VARIANT of type VT_EMPTY is returned. The function operates as shown in Table 8-22.

Table 8-22 WaitEvent parameters

Name	Type	Description
Event ID	LONG	Event ID for which to wait
Timeout	LONG	Timeout value to wait

The function returns a VARIANT.

8.3 Client Interface (IQXDM2)

This section describes the QXDM client interface (IQXDM2).

8.3.1 RequestItem

The RequestItem() function allows one or more requests to be scheduled for transmission to the target. The request itself is built according to a description entered into the QXDM database DIAG entity table. The function operates as shown in Table 8-23.

Table 8-23 RequestItem parameters

Name	Type	Description
pRequestName	BSTR	Name of DIAG entity (either a DIAG request or subsystem dispatch request)
pFields	BSTR	Fields used to build the request
bValuesOnly	VARIANT_BOOL	Does the pFields argument represent a string containing only the field values, or does it represent a fully qualified filename representing a text file containing the full field names and their values (such as generated by the QXDM Item Tester application)?
timeout	ULONG	Base timeout value in ms for request
requestCount	ULONG	Number of requests to schedule (must be at least one)
frequency	ULONG	If the requestCount argument is greater than one, this represents the amount of time (in ms) between successive requests.

The function returns a unique request ID. Upon error, 0 is returned.

NOTE If bValuesOnly is VARIANT_TRUE, then the format of pFields is a space-delimited string of field values where string field values are enclosed in quotes, i.e., 0xFF 100 “A String.”

This example specifies that the first field of the request has the value 0xFF, the second field has the value 100, and the last field for the request has the value A String. For fields that are enumerations, the raw value must be provided, i.e., the integral value.

NOTE If bValuesOnly is VARIANT_FALSE, then pFields is the fully qualified name of the file that gives the fields and their values. The format of the file is one field/value pair per line where the field is separated from the value by the ^ character. For an example, see the QXDM Item Tester application.

NOTE The timeout argument is dependent on the QPST Server. Current builds of QPST Server automatically retry each request up to three times in the case of a timeout while waiting for the response. Thus, the total timeout value is the argument value times three. Due to the real-time nature of QXDM, the maximum value for timeout is 2000 ms.

NOTE If the requestCount argument value is greater than one, then the request is considered to be repetitive. The first request will be scheduled to go out immediately. Subject to target latencies and the number of requests in the DIAG server queue, the next instance of this request will be sent out frequency ms after the response to the first request is received (or a timeout occurs). Due to the real-time nature of QXDM, the minimum value for the request frequency is currently 10 ms. A value of 0xFFFFFFFF for the request count indicates that the caller wants requests to be scheduled indefinitely.

8.3.2 RemoveRequest

The RemoveRequest() function allows one to cancel an existing request. The function operates as shown in Table 8-24.

Table 8-24 RemoveRequest parameters

Name	Type	Description
requestID	ULONG	Unique request ID

The function returns a VARIANT_BOOL indicating success/failure.

8.3.3 GetServerState

The GetServerState() function allows one to retrieve the state of the QXDM DIAG server. This is an extension of the previous IsConnected() function that takes no arguments and returns a ULONG. The ULONG return value is one of the values in Table 8-25.

Table 8-25 GetServerState return values

Value	Description
0	Target is disconnected. This is the initial state, which indicates that QXDM is not connected to a target.
1	QXDM is about to be connected to a target, QXDM has connected to a target via QPST Server, but all up-front initialization has not yet occurred.
2	Target has been verified, QXDM is connected to a target and has successfully completed all up-front initialization. This currently consists of querying the target for version information, supported log items, and supported message levels, etc.
3	QXDM is about to disconnect from a target. Commands must be sent to the target to get it into a desired state. Once those commands are sent, QXDM will transition to the Disconnected state.
4	In Playback mode, QXDM is not physically connected to a target. Rather, QXDM is operating from a file generated by a previous target connection. Async DIAG data is read from that file and distributed to interested parties as if connected to a live target. Note that playback can be reached only from the Disconnected state.
0xFFFFFFFF	Error

8.3.4 GetItemCount

The GetItemCount() function allows one to retrieve the number of items in the current QXDM Item Store. The function takes no arguments and returns a ULONG. The ULONG return value represents the total number of items in the Item Store (upon error 0 is returned).

8.3.5 GetItem

The GetItem() function allows one to retrieve a particular item from the current QXDM Item Store. The function operates as shown in Table 8-26.

Table 8-26 GetItem parameters

Name	Type	Description
index	ULONG	Index of item to retrieve (the Item Store is indexed 0 to the number of items returned by GetItemCount() – 1)

The function returns an interface pointer to the QXDM item interface model (IColorItem) as discussed in a subsequent section.

8.3.6 RegisterClient

The RegisterClient() function allows one to register as a client with QXDM. A client represents a subset of the QXDM Item Store whose contents are generally tailored to the needs of a particular display or script. The function operates as shown in Table 8-27.

Table 8-27 RegisterClient parameters

Name	Type	Description
pClientName	BSTR	Name of client or empty string (“”)
bFilteredView	VARIANT_BOOL	Bring up a QXDM filtered view tied to the client. If set to VARIANT_TRUE, a valid client name is required to be provided. As the filtered view is tied to the client, it will be restricted in functionality. The primary usage is intended to be for debugging user views and scripts.

The function returns a ULONG that represents the unique client identifier (handle). It is used in all subsequent client calls. Upon error 0xFFFFFFFF is returned.

8.3.7 RegisterQueueClient

The RegisterQueueClient() function allows one to register as a client with QXDM. A client represents a subset of the QXDM Item Store whose contents are generally tailored to the needs of a particular display or script. This version differs from RegisterClient() in that the client that is created internal to QXDM is of fixed size, i.e., only the last N items are available at any point in time. This allows for a more efficient representation that utilizes fewer system resources and supports expedited processing. The function operates as shown in Table 8-28.

Table 8-28 RegisterQueueClient parameters

Name	Type	Description
queueSize	ULONG	Size of queue (the value must be 256, 512, 1024, 2048, 4096, 16384, or 65536)

The function returns a ULONG that represents the unique client identifier (handle). It is used in all subsequent client calls. Upon error 0xFFFFFFFF is returned.

8.3.8 UnregisterClient

The UnregisterClient() function allows one to unregister an existing client. The function operates as shown in Table 8-29.

Table 8-29 UnregisterClient parameters

Name	Type	Description
hClient	ULONG	Unique handle of client to unregister

The function returns a VARIANT_BOOL indicating success/failure.

NOTE If a QXDM filtered view was associated with the client, the view will be closed automatically by the interface.

8.3.9 ConfigureClientByKeys

The ConfigureClientByKeys() function allows one to set the items an existing client is interested in processing. The function operates as shown in Table 8-30.

Table 8-30 ConfigureClientByKeys parameters

Name	Type	Description
hClient	ULONG	Unique handle of client to configure

The function returns an interface pointer to the QXDM client config interface model (IClientConfig) as discussed in a subsequent section.

8.3.10 ConfigureClientByNames

The ConfigureClientByNames() function allows one to set the items an existing client is interested in processing. The function operates as shown in Table 8-31.

Table 8-31 ConfigureClientByNames parameters

Name	Type	Description
hClient	ULONG	Unique handle of client to configure
pConfig	BSTR	Space-delimited names of DIAG entities in which client is interested in processing (each name must be in quotes)

The function returns a VARIANT_BOOL indicating success/failure.

NOTE The names come directly from the QXDM DIAG entity table. The format is a list of space-delimited strings in quotes, i.e., “EVENT_HDR_MSG_Rx” “1xEV-DO Active Set Pilot” “1xEV-DO State” “1xEV-DO Sector.”

This example indicates that the client wishes to process the event with the ID 329, and the logs with IDs 0x105F and 0x1080. When (or if) QXDM is connected to a target, it will configure the target to send the requested items. When an item arrives (either from the connected target or via QXDM playback) matching the given event or logs, it will then be added to the client.

8.3.11 ClearClientItems

The ClearClientItems() function allows one to empty an existing client of items. The function operates as shown in Table 8-32.

Table 8-32 ClearClientItems parameters

Name	Type	Description
hClient	ULONG	Unique handle of client to empty

The function returns a VARIANT_BOOL indicating success/failure.

NOTE If a QXDM filtered view was associated with the client, the view will be cleared automatically by the interface. In addition to this function, a client may be cleared when the current QXDM Item Store is emptied (as all clients are subsets of the current Item Store) or when an existing Item Store is loaded, replacing the current Item Store.

8.3.12 GetClientItemCount

The GetClientItemCount() function allows one to retrieve the number of items in the specified client. This will always be a number equal to or less than the number of items in the current QXDM Item Store. The function operates as shown in Table 8-33.

Table 8-33 GetClientItemCount parameters

Name	Type	Description
hClient	ULONG	Unique handle of client for which to obtain count

The function returns a ULONG. The ULONG return value represents the total number of items in the client (upon error 0 is returned).

8.3.13 GetClientItem

The GetClientItem() function allows one to retrieve a particular item from the specified client. The function operates as shown in Table 8-34.

Table 8-34 GetClientItem parameters

Name	Type	Description
hClient	ULONG	Unique handle of client from which to obtain item
index	ULONG	Index of item to retrieve (the client is indexed 0 to the number of items returned by GetClientItemCount() – 1).

The function returns an interface pointer to the QXDM item interface model (IColorItem) as discussed in a subsequent section.

8.3.14 ClientRequestItem

The ClientRequestItem() function allows one to schedule one or more requests to be sent through the specified client. The request itself is built according to a description entered into the QXDM database DIAG entity table. The function operates as shown in Table 8-35.

Table 8-35 ClientRequestItem parameters

Name	Type	Description
hClient	ULONG	Unique handle of client through which to request item
pRequestName	BSTR	Name of DIAG entity (either a DIAG request or subsystem dispatch request)
pFields	BSTR	Fields used to build the request
bValuesOnly	VARIANT_BOOL	Does the pFields argument represent a string containing only the field values or does it represent a fully qualified file name representing a text file containing the full field names and their values (such as generated by the QXDM Item Tester application)?
timeout	ULONG	Base timeout value in milliseconds for request
requestCount	ULONG	Number of requests to schedule (must be at least one)
frequency	ULONG	If the requestCount argument is greater than one, then this represents the amount of time (in milliseconds) between successive requests.

The function returns a unique request ID. Upon error, 0 is returned.

NOTE If bValuesOnly is VARIANT_TRUE, then the format of pFields is a space-delimited string of field values where string field values are enclosed in quotes, i.e., 0xFF 100 “A String.” This example specifies that the first field of the request has the value 0xFF, the second field has the value 100, and the last field for the request has the value A String. For fields that are enumerations, the raw value must be provided, i.e., the integral value.

NOTE If bValuesOnly is VARIANT_FALSE, then pFields is the fully qualified name of the file that gives the fields and their values. The format of the file is one field/value pair per line where the field is separated from the value by the ^ character. For an example, see the QXDM Item Tester application.

NOTE The timeout argument is dependent on QPST Server. Current builds of QPST Server automatically retry each request up to three times in the case of a timeout while waiting for the response. Thus, the total timeout value is the argument value times three. Due to the real-time nature of QXDM, the maximum value for timeout is 2000 ms.

NOTE If the requestCount argument value is greater than one, the request is considered to be repetitive. The first request will be scheduled to go out immediately. Subject to target latencies and the number of requests in the DIAG server queue, the next instance of this request will be sent out frequency ms after the response to the first request is received (or a timeout occurs). Due to the real-time nature of QXDM, the minimum value for the request frequency is currently 10 ms. A value of 0xFFFFFFFF for the request count indicates that the caller wants requests to be scheduled indefinitely.

NOTE Requests and the responses to them are automatically added to the client. Thus, there is no need to register for each request and response type. Due to this it is important that the client exist for the duration of the request/response cycle. Although QXDM will continue to function if the client is removed (unregistered) prior to the request being transmitted or the response being received, relying on this behavior is not suggested.

NOTE For subsystem dispatch V2 items, only the immediate response will appear in the client view. Registration for the delayed response is required for the client to receive any actual delayed response packets (see Section 8.4.7).

8.3.15 ClientRemoveRequest

The ClientRemoveRequest() function allows one to cancel an existing request. The function operates as shown in Table 8-36.

Table 8-36 ClientRemoveRequest parameters

Name	Type	Description
hClient	ULONG	Unique handle of client
requestID	ULONG	Unique request ID

The function returns a VARIANT_BOOL indicating success/failure.

8.3.16 ClientRequestNVRead

The ClientRequestNVRead() function allows one to schedule a single NV item read request to be sent through the specified client. The request itself is built according to a description entered into the QXDM database DIAG entity table. The function operates as shown in Table 8-37.

Table 8-37 ClientRequestNVRead parameters

Name	Type	Description
hClient	ULONG	Unique handle of client through which to request item
pNVItemName	BSTR	Name of DIAG entity to request (entity must be an NV item)
pFields	BSTR	Fields used to build the request
bValuesOnly	VARIANT_BOOL	Does the pFields argument represent a string containing only the field values or does it represent a fully qualified filename representing a text file containing the full field names and their values (such as generated by the QXDM Item Tester application)
timeout	ULONG	Base timeout value in ms for request

The function returns a unique request ID. Upon error, 0 is returned.

NOTE This function behaves in a manner consistent with the ClientRequestItem() function.

8.3.17 ClientRequestNVWrite

The ClientRequestNVWrite() function allows one to schedule a single NV item write request to be sent through the specified client. The request itself is built according to a description entered into the QXDM database DIAG entity table. The function operates as shown in Table 8-38.

Table 8-38 ClientRequestNVRead parameters

Name	Type	Description
hClient	ULONG	Unique handle of client through which to request item
pNVItemName	BSTR	Name of DIAG entity to request (entity must be an NV item)
pFields	BSTR	Fields used to build the request
bValuesOnly	VARIANT_BOOL	Does the pFields argument represent a string containing only the field values or does it represent a fully qualified file name representing a text file containing the full field names and their values (such as generated by the QXDM Item Tester application)?
timeout	ULONG	Base timeout value in milliseconds for request

The function returns a unique request ID. Upon error 0 is returned.

NOTE This function behaves in a manner consistent with the ClientRequestItem() function.

8.3.18 CopyClientItems

The CopyClientItems() function allows one to copy all or a portion of the items of a client to a new Item Store Format file. The function operates as shown in Table 8-39.

Table 8-39 CopyClientItems parameters

Name	Type	Description
hClient	ULONG	Unique handle of client from which to copy items
pFileName	BSTR	Filename of new Item Store Format file (.ISF)
startIndex	ULONG	Client index of first item to copy
endIndex	ULONG	Client index of last item to copy

The function returns a VARIANT_BOOL indicating overall success/failure.

The index parameters give the range of items to copy. The start index must be less than or equal to the end index. If the end index is greater than the number of items in the client, the actual index used will be one minus the number of items in the client, i.e., to always copy the entire contents of a client to the new file store without regard to the current size of the client, (0, 0xFFFFFFFF) should be specified.

NOTE This operation may take a nontrivial amount of time to complete, depending on the number of items managed by the client and the parameters to the function. While it executes, the function will not return. However, the QXDM interface will continue to respond to additional commands. Due to this, care must be taken in ensuring that the client is neither unregistered nor emptied asynchronously while a copy-items operation on the client is executing.

8.3.19 SyncToItem

The SyncToItem() function allows one to sync item list-based views to a given item index. The function operates as shown in Table 8-40.

Table 8-40 SyncToItem parameters

Name	Type	Description
Index	ULONG	Item Store index of the item to sync to
bSyncExact	VARIANT_BOOL	If true, sync to the exact index; otherwise, sync to the nearest item

The function returns a VARIANT_BOOL indicating overall success/failure.

8.3.20 SyncToClientItem

The SyncToClientItem() function allows one to sync a client item list-based view to a given item index. The function operates as shown in Table 8-41.

Table 8-41 SyncToClientItem parameters

Name	Type	Description
hClient	ULONG	Unique handle of client view
Index	ULONG	Client index of the item to sync to
bSyncExact	VARIANT_BOOL	If true, sync to the exact index; otherwise, sync to the nearest item

The function returns a VARIANT_BOOL indicating overall success/failure.

8.4 Client Config Interface (IClientConfig)

This section describes the QXDM client config interface (IClientConfig).

8.4.1 ClearConfig

The ClearConfig() function allows one to clear a client config. In the Cleared state, a client will only receive items originated through the ClientRequestItem(), ClientRequestNVRead(), and ClientRequestNVWrite() functions. This function takes no arguments and returns no value.

8.4.2 CommitConfig

The CommitConfig () function is typically the last function called when adjusting the config of a client. All changes made to the client config are updated within QXDM when CommitConfig() is called. This function takes no arguments and returns no value.

8.4.3 AddItem

The AddItem() function allows one to request that items of the specified type are added to the client as they are processed by the current QXDM Item Store. The function operates as shown in Table 8-42 and does not return a value.

Table 8-42 AddItem parameters

Name	Type	Description
itemType	ULONG	Desired item type, possible values are: ■ 0 – Malformed DIAG entity ■ 1 – Target DIAG response (minus following) ■ 2 – Target DIAG request (minus following) ■ 3 – GPS information ■ 4 – Target event ■ 5 – Target log ■ 6 – Target message ■ 7 – Generic strings ■ 8 – Target OTA log ■ 9 – Target subsystem dispatch response ■ 10 – Target subsystem dispatch request

NOTE Contrary to subsequent functions that request items of a particular subtype, this function does not actively register with the target for the given items.

NOTE This form of registration is equivalent to checking an Item Type from the Item List Config of a QXDM Filtered View.

8.4.4 AddDIAGRequest

The AddDIAGRequest() function allows one to request that outgoing DIAG requests with a specified DIAG command code be added to the client. The function operates as shown in Table 8-43 and does not return a value.

Table 8-43 AddDIAGRequest parameters

Name	Type	Description
cmd	ULONG	DIAG command code of items in which client is interested

NOTE This is not the same as building and scheduling a request via ClientRequestItem(). This simply adds outgoing DIAG requests of a particular command code to the client as they are processed by the Item Store. The actual request may have originated from another QXDM subsystem.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select DIAG Requests from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular DIAG request command code from the subtype list (such as Version Information Request).

In the above example, all DIAG requests with command code 0 would be added to the client regardless of where they originated.

8.4.5 AddDIAGResponse

The AddDIAGResponse() function allows one to request that incoming DIAG responses with a specified DIAG command code be added to the client. The function operates as shown in Table 8-44 and does not return a value.

Table 8-44 AddDIAGResponse parameters

Name	Type	Description
cmd	ULONG	DIAG command code of items in which client is interested

NOTE This is not the same as building and scheduling a request via ClientRequestItem(). This simply adds incoming DIAG responses of a particular command code to the client as they are processed by the Item Store. The actual response may have originated from a DIAG request made by another QXDM subsystem or from QXDM playback.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select DIAG Requests from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular DIAG request command code from the subtype list (such as Version Information Request).

In the above example, all DIAG requests with command code 0 would be added to the client regardless of where they originated.

8.4.6 AddSubsysRequest

The AddSubsysRequest() function allows one to request that outgoing DIAG subsystem dispatch requests with a specified DIAG subsystem ID and subsystem command code be added to the client. The function operates as shown in Table 8-45 and does not return a value.

Table 8-45 AddSubsysRequest parameters

Name	Type	Description
subsysID	ULONG	DIAG subsystem dispatch ID of items in which client is interested
subsysyCmd	ULONG	DIAG subsystem dispatch command code of items in which client is interested

NOTE This is not the same as building and scheduling a request via ClientRequestItem(). This adds outgoing DIAG subsystem dispatch requests of a particular ID and command code to the client as they are processed by the Item Store. The actual request may have originated from another QXDM subsystem.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Subsystem Dispatch Requests from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular subsystem ID and command code from the subtype list (such as WCDMA/Version Request).

In the above example, all DIAG subsystem dispatch requests with subsystem ID 4 and subsystem command code 0 would be added to the client regardless of where they originated.

8.4.7 AddSubsysResponse

The AddSubsysResponse() function allows you to request that incoming DIAG subsystem dispatch responses with a specified DIAG subsystem ID and subsystem command code be added to the client. The function operates as shown in Table 8-46 and does not return a value.

Table 8-46 AddSubsysResponse parameters

Name	Type	Description
subsysID	ULONG	DIAG subsystem dispatch ID of items in which client is interested
subsysyCmd	ULONG	DIAG subsystem dispatch command code of items in which client is interested

NOTE This is not the same as building and scheduling a request via ClientRequestItem(). This adds outgoing DIAG subsystem dispatch requests of a particular ID and command code to the client as they are processed by the Item Store. The actual response may have originated from a request made by another QXDM subsystem or from QXDM playback.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Subsystem Dispatch Responses from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular subsystem ID and command code from the subtype list (such as WCDMA/Version Response).

In the above example, all DIAG subsystem dispatch responses with subsystem ID 4 and subsystem command code 0 would be added to the client regardless of where they originated.

8.4.8 AddEvent

The AddEvent() function allows one to request that incoming events with a specified event ID be added to the client. The function operates as shown in Table 8-47 and does not return a value.

Table 8-47 AddEvent parameters

Name	Type	Description
eventID	ULONG	Event ID of items in which client is interested

NOTE This function results in the on-target event mask (if supported) being adjusted to include the specified event ID.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Event Reports from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular event ID from the subtype list (such as EVENT_BAND_CLASS_CHANGE).

In the above example, all events with ID 256 would be added to the client regardless of where they originated.

8.4.9 AddLog

The AddLog() function allows one to request that incoming logs with a specified log code be added to the client. The function operates as shown in Table 8-48 and does not return a value.

Table 8-48 AddLog parameters

Name	Type	Description
logCode	ULONG	Log code of items in which client is interested

NOTE This function results in the appropriate on-target log mask being adjusted to include the specified log code. Also note that QXDM breaks out specific OTA log packets in order to provide additional functionality. Attempting to register from OTA log packets using AddLog() will not produce the desired behavior.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Log Packets from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular log code from the subtype list (such as General Temporal Analyzer).

In the above example, all logs with log code 0x1019 would be added to the client regardless of where they originated.

8.4.10 AddMessage

The AddMessage() function allows one to request that incoming debug trace messages with a specified subsystem ID and message level be added to the client. The function operates as shown in Table 8-49 and does not return a value.

Table 8-49 AddLog parameters

Name	Type	Description
subsysID	ULONG	Subsystem ID of items in which client is interested
level	ULONG	Message level (under above subsystem ID) of items in which client is interested

NOTE This function results in the appropriate on-target message mask being adjusted to include the specified debug message.

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Message Packets from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular subsystem ID and message level from the subtype list (such as Legacy/Fatal).

In the above example, all debug messages with subsystem ID 0 and message level 4 would be added to the client regardless of where they originated.

8.4.11 AddString

The AddString() function allows one to request that string items of the specified subtype are added to the client as they are processed by the current QXDM Item Store. The function operates as shown in Table 8-50 and does not return a value.

Table 8-50 AddString parameters

Name	Type	Description
stringType	ULONG	Desired string item sub-type, possible values are: ■ 0 – Informational string ■ 1 – Warning string ■ 2 – Error string ■ 3 – Automation string ■ 4 – Connection state string

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Strings from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular string subtype from the subtype list (such as Automation).

In the above example, all strings originating from the QXDM automation subsystem (or QXDM playback of an automation string) will be added to the client.

8.4.12 AddOTALog

The AddOTALog() function allows one to request that incoming OTA logs with a specified log code and message type key be added to the client. The function operates as shown in Table 8-51 and does not return a value.

Table 8-51 AddOTALog parameters

Name	Type	Description
logCode	ULONG	Log code of items in which client is interested
key	VARIANT (VT_ARRAY | VT_UI4)	Message type key (specific to log code)

NOTE This function results in the appropriate on-target log mask being adjusted to include the specified log code.

QXDM currently treats the log codes in Table 8-52 through Table 8-57 as OTA logs. The composition of each key is provided as well.

Table 8-52 OTA log codes/keys (CDMA)

Code	Description	Key
0x1004	CDMA access channel	Single-entry array (message type)
0x1005	CDMA reverse traffic channel	Single-entry array (message type)
0x1006	CDMA sync channel	Single-entry array (message type)
0x1007	CDMA paging channel	Single-entry array (message type)
0x1008	CDMA forward traffic channel	Single-entry array (message type)
0x1090	CDMA forward dedicated control channel	Single-entry array (message type)
0x1091	CDMA reverse dedicated control channel	Single-entry array (message type)
0x10D6	CDMA forward broadcast control channel	Single-entry array (message type)
0x10D7	CDMA reverse enhanced access channel	Single-entry array (message type)
0x10D8	CDMA forward common control channel	Single-entry array (message type)
0x1388	CDMA (from UMTS) paging channel	Single-entry array (message type)
0x1389	CDMA (from UMTS) forward traffic channel	Single-entry array (message type)
0x138B	CDMA (from VoIP) paging channel	Single-entry array (message type)
0x138C	CDMA (from VoIP) access channel	Single-entry array (message type)
0x138D	CDMA (from VoIP) forward traffic channel	Single-entry array (message type)

Table 8-53 OTA log codes/keys (CDMA 1xEV-DO)

Code	Description	Key
0x1076	1xEV-DO access channel	Two-entry array (protocol, message type)
0x1077	1xEV-DO reverse traffic channel	Two-entry array (protocol, message type)
0x1078	1xEV-DO directed control channel	Two-entry array (protocol, message type)
0x1079	1xEV-DO forward traffic channel	Two-entry array (protocol, message type)
0x107C	1xEV-DO broadcast control channel	Two-entry array (protocol, message type)
0x1085	1xEV-DO MAC access channel	Empty
0x1086	1xEV-DO MAC control channel	Empty

Table 8-54 OTA log codes/keys (GSM)

Code	Description	Key
0x512F	GSM radio resource	Two-entry array (channel/direction, message type)
0x5226	GSM medium access control	Two-entry array (channel/direction, message type)
0x5230	GSM session/mobility management	Two-entry array (direction, message type)

Table 8-55 OTA log codes/keys (UMTS)

Code	Description	Key
0x713A	UMTS NAS layer	Two-entry array (protocol, message type)

Table 8-56 OTA log codes/keys (WCDMA)

Code	Description	Key
0x412F	WCDMA radio resource control	Two-entry array (channel, message type)

NOTE The following table was added to this document revision.

Table 8-57 OTA log codes/keys (Bluetooth)

Code	Description	Key
0x1048	Bluetooth LMP Rx	Empty
0x1049	Bluetooth LMP Tx	Empty
0x104C	Bluetooth L2CAP Rx	Empty
0x104D	Bluetooth L2CAP Tx	Empty
0x1052	Bluetooth RFCOMM Rx	Empty
0x1053	Bluetooth RFCOMM Tx	Empty

NOTE This form of registration is equivalent to the following actions in a QXDM Filtered View:

1. Bring up the Item List Config dialog box.
2. Check and select Log Packets (OTA) from the Item Types list.
3. Check Filter/Register On Target For Items.
4. Select a particular log code and message type from the subtype list (such as Access/Registration).

In the above example, all OTA logs with log code 0x1004 and message type 1 would be added to the client regardless of where they originated.

8.4.13 AddOTALogByString

The AddOTALogByString() function allows one to request that incoming OTA logs with a specified log code and message type key be added to the client. The function operates as shown in Table 8-58 and does not return a value.

Table 8-58 AddOTALogByString parameters

Name	Type	Description
logCode	ULONG	Log code of items in which client is interested
pKey	BSTR	Message type key (specific to log code) expressed as a string of space-delimited values

NOTE This function operates in the same fashion as AddOTALog(). The sole difference is that the message type key is passed as a string consisting of space-delimited values, i.e., “5 6 7”). The intended use is in scripting languages that do not easily support constructing and manipulating VARIANT arrays of simple types, such as VB Script and JScript.

8.4.14 SetSubsysV2DelayedResponsesOnly

The SetSubsysV2DelayedResponsesOnly() function allows one to configure a client that will receive only the delayed subsystem dispatch V2 responses (as opposed to both the immediate and delayed responses). Since both V2 and V1 responses are treated as the same item type, this makes certain types of filtering and item processing more convenient. The function operates as shown in Table 8-59 and does not return a value.

Table 8-59 SetSubsysV2DelayedResponsesOnly

Name	Type	Description
bDelayedOnly	VARIANT_BOOL	Accept only delayed responses if true

8.5 Item Interface (IColorItem)

This section describes the QXDM item interface (IColorItem).

8.5.1 GetItemType

The GetItemType() function allows one to retrieve the item type of the current color item. The function takes no arguments and returns a ULONG value. The ULONG return value is one of the values shown in Table 8-60.

Table 8-60 GetItemType return values

Value	Description
0	Malformed DIAG entity
1	Target DIAG response (minus following)
2	Target DIAG request (minus following)
3	GPS information
4	Target event
5	Target log
6	Target message
7	Generic strings
8	Target OTA log
9	Target subsystem dispatch response
10	Target subsystem dispatch request
0xFFFFFFFF	Error

NOTE These are the same values (excluding the final error value) used as arguments to the AddItem() function.

8.5.2 GetItemTypeText

The GetItemTypeText() function allows one to retrieve the item type converted to a string. The function takes no arguments and returns a BSTR value.

NOTE The string returned will be identical to the contents of the Type column in any QXDM list-based view.

8.5.3 GetItemColor

The GetItemColor() function allows one to retrieve the color of the current color item. The function takes no arguments and returns a ULONG value. The ULONG return value is in the form of a standard Windows COLORREF value, i.e., three 8-bit values (R, G, B) packed into a 32-bit space. Upon return, (0, 0, 0) is returned. This represents black, an invalid value in QXDM, as that is the color of all item list backgrounds.

8.5.4 GetItemTimestamp

The GetItemTimestamp() function allows one to retrieve the generic timestamp of the current color item. The function takes no arguments and returns a standard Windows DATE value (VT_DATE).

NOTE The generic timestamp represents the time the color item was created, i.e., when it first entered the current Item Store. This is not the same as the item-specific timestamp, which is assigned to the item by the target.

8.5.5 GetItemTimestamp2

The GetItemTimestamp2() function allows one to retrieve the generic timestamp of the current color item. The function takes no arguments and returns a standard Windows DATE value as a double precision floating point number (VT_R8).

NOTE The generic timestamp represents the time the color item was created, i.e., when it first entered the current Item Store. This is not the same as the item-specific timestamp, which is assigned to the item by the target.

8.5.6 GetItemTimestampText

The GetItemTimestampText() function allows one to retrieve the generic item timestamp converted to a string. The function operates as shown in Table 8-61. The function returns a BSTR.

Table 8-61 GetItemTimestampText parameters

Name	Type	Description
bWantDate	VARIANT_BOOL	Include the date
bWantMillisecs	VARIANT_BOOL	Include ms

NOTE The string returned will be identical to the contents of the Timestamp column in any QXDM list-based view.

8.5.7 GetItemSpecificTimestamp

The GetItemSpecificTimestamp() function allows one to retrieve the item-specific timestamp of the current color item. The function takes no arguments and returns a standard Windows DATE value (VT_DATE).

NOTE The item-specific timestamp represents the time assigned to the item by the target. If the item does not support a target-assigned timestamp, then the generic timestamp will be returned instead.

8.5.8 GetItemSpecificTimestamp2

This function allows one to retrieve the item-specific timestamp of the current color item. The function takes no arguments and returns a standard Windows DATE value as a double precision floating point number (VT_R8).

NOTE The item-specific timestamp represents the time assigned to the item by the target. If the item does not support a target-assigned timestamp, then the generic timestamp will be returned instead.

8.5.9 GetItemSpecificTimestampText

The GetItemSpecificTimestampText() function allows one to retrieve the item-specific timestamp converted to a string. The function operates as shown in Table 8-62.

Table 8-62 GetItemSpecificTimestampText parameters

Name	Type	Description
bWantDate	VARIANT_BOOL	Include the date
bWantMillisecs	VARIANT_BOOL	Include milliseconds

A BSTR value is returned.

NOTE The string returned will be identical to the contents of the Timestamp column in any QXDM list-based view.

8.5.10 GetItemBuffer

The GetItemBuffer() function allows one to retrieve the raw buffer of the current color item. The function operates as shown in Table 8-63.

Table 8-63 GetItemBuffer parameters

Name	Type	Description
bIncludeHeaderBytes	VARIANT_BOOL	Return the full item buffer, including the QXDM-defined header or simply the QXDM-defined payload

The function returns a VARIANT array (VT_ARRAY | VT_UI1) when successful. When an error occurs, an empty VARIANT array is returned. However, as the QXDM-defined payload may itself be empty, this is not a reliable way of determining if a true error occurred.

8.5.11 GetItemDLFBuffer

The GetItemDLFBuffer() function allows one to retrieve the raw buffer of the current color item adjusted to be compatible with the legacy DLF log file format.

The function returns a VARIANT array (VT_ARRAY | VT_UI1) when successful. When an error occurs, an empty VARIANT array is returned.

8.5.12 GetItemBufferText

The GetItemBufferText() function allows one to retrieve the raw buffer of the current color item converted to a string. The function returns a BSTR. The function operates as shown in Table 8-64.

Table 8-64 GetItemBufferText parameters

Name	Type	Description
bIncludeHeaderBytes	VARIANT_BOOL	Return the full item buffer, including the QXDM-defined header or simply the QXDM-defined payload

NOTE The string returned will be identical to the contents of the Payload column in any QXDM list-based view.

8.5.13 GetItemKeyText

The GetItemKeyText() function allows one to retrieve the item key converted to a string. The function takes no arguments and returns a BSTR value.

NOTE The string returned will be identical to the contents of the Key column in any QXDM list-based view.

8.5.14 GetItemName

The GetItemName() function allows one to retrieve the item name. The function takes no arguments and returns a BSTR value.

NOTE The string returned will be identical to the contents of the Name column in any QXDM list-based view. The item name represents the item key mapped to a string defined in the appropriate QXDM database category/reference table.

8.5.15 GetItemSummary

The GetItemNameSummary() function allows one to retrieve the item summary. The function takes no arguments and returns a BSTR value.

NOTE The string returned will be identical to the contents of the Summary column in any QXDM list-based view. The item summary represents a summary of the item payload generated either by internal QXDM processes, the QXDM database structure definition in combination with a QXDM database format specifier, or an external dynamic item parsing DLL.

8.5.16 GetItemSize

The GetItemSize() function allows one to retrieve the item size of the current color item. The function returns a ULONG value representing the size in bytes of the item. The function operates as shown in Table 8-65.

Table 8-65 GetItemSize parameters

Name	Type	Description
bIncludeHeaderBytes	VARIANT_BOOL	Return the full item size, including the QXDM-defined header or simply the QXDM-defined payload

8.5.17 GetItemParsedText

The GetItemParsedText() function allows one to retrieve the item full parsed text. The function takes no arguments and returns a BSTR value.

NOTE The string returned will be identical to the contents of the bottom right pane in any QXDM list-based view (currently excludes the database parsed field list). The item full parsed text is a full description of the item payload generated either by internal QXDM processes, the QXDM database structure definition, an external dynamic item parsing DLL, or (in the case of OTA logs) SILK.

8.5.18 GetItemFieldValue

The GetItemFieldValue() function (Table 8-66) allows one to retrieve the actual value of a field parsed out of the item buffer. Typical applications of this function would be to parse items that cannot be described by the database or to extract fields out of the header of an item. The function operates as shown in Table 8-66.

Table 8-66 GetItemFieldValue parameters

Name	Type	Description
fieldType	ULONG	Type of field to obtain (see Table 8-67)
bitCount	ULONG	Number of bits to extract
bitOffset	ULONG	Offset into item buffer to begin extraction
bIncludeHeaderBytes	VARIANT_BOOL	Include the header in item buffer

The function returns a VARIANT. In the case of an error, an empty VARIANT (VT_EMPTY) is returned.

Table 8-67 shows the types of supported fields.

Table 8-67 Supported field types

Type	Value	VARIANT type	Description
BOOL	0	VT_BOOL	Boolean (true/false)
INT8	1	VT_I1	8-bit signed value
UINT8	2	VT_UI1	8-bit unsigned value
INT16	3	VT_I2	16-bit signed value
UINT16	4	VT_UI2	16-bit unsigned value
INT32	5	VT_I4	32-bit signed value
UINT32	6	VT_UI4	32-bit unsigned value
INT64	7	VT_I8	64-bit signed value
UINT64	8	VT_UI8	64-bit unsigned value
ANSI String	9	VT_BSTR	ANSI string (fixed length given by bitCount)
UNICODE String	10	VT_BSTR	UNICODE string (fixed length given by bitCount)
ANSI String	11	VT_BSTR	ANSI string (NULL terminated, bitCount ignored)
UNICODE String	12	VT_BSTR	UNICODE string (NULL terminated, bitCount ignored)
FLOAT32	13	VT_R4	IEEE 32-bit floating point value
FLOAT64	14	VT_R8	IEEE 64-bit floating point value

8.5.19 GetItemFieldValueText

The GetItemFieldValueText() function allows one to retrieve the text representation of the field value parsed out of the item buffer. The function operates as shown in Table 8-68.

Table 8-68 GetItemFieldValueText parameters

Name	Type	Description
fieldType	ULONG	Type of field to obtain (see Table 8-67)
bitCount	ULONG	Number of bits to extract
bitOffset	ULONG	Offset into item buffer to begin extraction
bIncludeHeaderBytes	VARIANT_BOOL	Include the header in item buffer
bHexadecimal	VARIANT_BOOL	Format field text as hexadecimal

The function returns the field value as a VT_BSTR and operates in the same manner as GetItemFieldValue described above.

8.5.20 GetItemFields

The GetItemFields() function allows one to retrieve the QXDM structure database representation of the current color item. The function takes no arguments and returns an interface pointer to the QXDM database parsed field interface model (IColorItemFields) as discussed in Section 8.6.

8.5.21 GetNamedItemFields

The GetNamedItemFields() function allows one to retrieve the QXDM structure database representation of the current color item. The function takes a single argument (described in Table 8-69) and returns an interface pointer to the QXDM database parsed field interface model (IColorItemFields) as discussed in Section 8.6.

Table 8-69 GetNamedItemFields parameters

Name	Type	Description
pEntityName	BSTR	Unique DIAG entity name

NOTE The function operates in a manner consistent with GetItemFields() with the exception that a unique DIAG entity name is passed by the caller. The name identifies/selects the structure definition that is used to parse the current item. If the name is empty, then the function operates exactly as GetItemFields(), i.e., the default structure definition is used. The QXDM database defines default as the first structure definition added to the database for a particular DIAG entity key. Since the QXDM database is always loaded before any user database, this implies that the structure definition (when present) in the QXDM database is the default. Should the QXDM database not define a structure for a given DIAG entity key, then the default is dependant on the contents and load order of the user databases. The current QXDM behavior is for the load order to be alphabetical by user database name. Note that this behavior may change in the future.

8.5.22 GetConfiguredItemFields

The GetConfiguredItemFields() function allows one to retrieve the database parsed item fields of the current color item. For applications that process a large number of items per sec or process items that parse to a large number of fields, setting bFieldStrings and/or bFieldNames to false will improve performance and/or is usually dependant on whether you need field strings or not. Generally, you only need field strings if you are dealing with a structure that contains enumerations and you need to display the mapped enumeration value strings. The function operates as shown in Table 8-70.

Table 8-70 GetConfiguredItemFields parameters

Name	Type	Description
pEntityName	LPCTSTR	DIAG entity name (if this is not given then the default structure associated with this entity will be used, i.e., GetItemFields() behavior)
bFieldStrings	VARIANT_BOOL	Generate string representations of field values
bFieldNames	VARIANT_BOOL	Generate partial field names

The function returns an interface pointer to the QXDM database parsed field interface model, IColorItemFields (Section 8.6).

NOTE If bFieldStrings is set to false, retrieving the string representation of fields will not be available. If bFieldNames is set to false, searching for the index of a field by name will not be available.

8.6 Field Interface (IColorItemFields)

This section describes the QXDM database parsed field interface (IColorItemFields).

8.6.1 GetXML

The GetXML() function allows one to retrieve the database parsed fields as an XML-formatted string. The function takes no arguments and returns a BSTR value.

NOTE The format of the XML is given in the following (partial) example:

```
<Entity>
  <Name>Searcher And Finger</Name>
  <Fields>
    <Field>
      <NamePartial>Transmit Diversity Pilot</NamePartial>
      <ValueText>0</ValueText>
      <ValueRaw>0</ValueRaw>
      <BitSize>1</BitSize>
      <BitOffset>0</BitOffset>
    </Field>
  </Fields>
</Entity>
```

The intended usage of this function is in crafting simpler HTML-based QXDM views using XML transformations. In some cases, there may be an additional performance increase when using XML, as it should result in fewer COM interface classes than iterating over the list of fields.

8.6.2 GetFieldCount

The GetFieldCount() function allows one to retrieve the number of parsed fields generated by the database parser. The function takes no arguments and returns a ULONG value.

8.6.3 GetFieldIndex

The GetFieldIndex() function allows one to retrieve the index of the first field matching the specified field name. The function operates as shown in Table 8-71.

Table 8-71 GetFieldIndex parameters

Name	Type	Description
pName	BSTR	Name (either fully qualified or partial) of the field for which to find the index
bPartial	VARIANT_BOOL	Is the first argument a partial field name?

The function returns a ULONG that represents the index into the parsed field list matching the specified field name. The field list itself is indexed 0 up to but not including the value returned by GetFieldCount(). In the case of an error, the value 0xFFFFFFFF is returned.

NOTE For the purposes of reducing search time, subsequent calls start from one beyond the last index returned by this method. If the provided field name is not found before reaching the end of the field list, the search wraps around to the first field and proceeds until the provided field name is found or the last index returned is reached.

8.6.4 GetFieldIndexFrom

The GetFieldIndexFrom() function allows one to retrieve the index of the first field matching the specified field name starting the search from the provided field index. The function operates as shown in Table 8-72.

Table 8-72 GetFieldIndexFrom parameters

Name	Type	Description
pName	BSTR	Name (either fully qualified or partial) of the field for which to find the index
startIndex	ULONG	Index of field to start the search from
bPartial	VARIANT_BOOL	Is the first argument a partial field name?
bLoop	VARIANT_BOOL	If the field name is not found before reaching the end of the field list, loop back to beginning of the field list?

The function returns a ULONG that represents the index into the parsed field list matching the specified field name.

NOTE The index returned affects the start index of GetFieldIndex() as described above.

8.6.5 GetFieldIndexByID

The GetFieldIndexByID() function allows one to retrieve the index of the first field matching the specified field ID. The field ID is the identifier assigned to the field when it added to the DM database. The function operates as shown in Table 8-73.

Table 8-73 GetFieldIndexByID parameters

Name	Type	Description
fieldID	ULONG	ID of the field for which to find the index

The function returns a ULONG that represents the index into the parsed field list matching the specified field ID. The field list itself is indexed 0 up to but not including the value returned by GetFieldCount(). In the case of an error, the value 0xFFFFFFFF is returned.

NOTE For the purposes of reducing search time, subsequent calls start from one beyond the last index returned by this method. If the provided field name is not found before reaching the end of the field list, the search wraps around to the first field and proceeds until the provided field name is found or the last index returned is reached.

8.6.6 GetFieldIndexFromByID

The GetFieldIndexFromByID() function allows one to retrieve the index of the first field matching the specified field ID starting the search from the provided field index. The function operates as shown in Table 8-74.

Table 8-74 GetFieldIndexFromByID parameters

Name	Type	Description
fieldID	ULONG	ID of the field for which to find the index
startIndex	ULONG	Index of field to start the search from
bLoop	VARIANT_BOOL	If the field name is not found before reaching the end of the field list, loop back to beginning of the field list?

The function returns a ULONG that represents the index into the parsed field list matching the specified field ID. The field list itself is indexed 0 up to but not including the value returned by GetFieldCount(). In the case of an error, the value 0xFFFFFFFF is returned.

NOTE The index returned affects the start index of GetFieldIndexByID() as described above.

8.6.7 GetFieldOffset

The GetFieldOffset() function allows one to retrieve the bit offset of the field identified by the given field index. The function operates as shown in Table 8-75.

Table 8-75 GetFieldOffset parameters

Name	Type	Description
index	ULONG	Field index

The function returns a ULONG representing the bit offset from the start of the QXDM-defined item payload of the specified field. In the case of an error, the value 0xFFFFFFFF is returned.

8.6.8 GetFieldSize

The GetFieldSize() function allows one to retrieve the bit size of the field identified by the given field index. The function operates as shown in Table 8-76.

Table 8-76 GetFieldSize parameters

Name	Type	Description
index	ULONG	Field index

The function returns a ULONG. In the case of an error, the value 0xFFFFFFFF is returned.

8.6.9 GetFieldValue

The GetFieldValue() function allows one to retrieve the raw value of the field identified by the given field index. The function operates as shown in Table 8-77.

Table 8-77 GetFieldValue parameters

Name	Type	Description
index	ULONG	Field index

The function returns a VARIANT. In the case of an error, an empty VARIANT (VT_EMPTY) is returned. The QXDM structure database supports the destination field types shown below.

Table 8-78 Supported field types

Type	VARIANT Type	Description
BOOL	VT_BOOL	Boolean (true/false)
INT8	VT_I1	8-bit signed value
UINT8	VT_UI1	8-bit unsigned value
INT16	VT_I2	16-bit signed value
UINT16	VT_UI2	16-bit unsigned value
INT32	VT_I4	32-bit signed value
UINT32	VT_UI4	32-bit unsigned value
INT64	VT_I8	64-bit signed value
UINT64	VT_UI8	64-bit unsigned value
ANSI String	VT_BSTR	ANSI string
UNICODE String	VT_BSTR	UNICODE string
FLOAT32	VT_R4	IEEE 32-bit floating point value
FLOAT64	VT_R8	IEEE 64-bit floating point value
Signed Enum	VT_I4	Signed enumeration
Unsigned Enum	VT_UI4	Unsigned enumeration

8.6.10 GetFieldName

The GetFieldName() function allows one to name the field identified by the given field index. The function operates as shown in Table 8-79.

Table 8-79 GetFieldName parameters

Name	Type	Description
index	ULONG	Field index
bPartial	VARIANT_BOOL	Return partial field name?

The function returns a BSTR representing the field name.

NOTE A partial field name is equivalent to a fully qualified field name minus the leading DIAG entity name and field delimiter (‘.’).

8.6.11 GetFieldValueText

The GetFieldValueText() function allows one to retrieve the value of the field identified by the given field index expressed as a string. The function operates as shown in Table 8-80.

Table 8-80 GetFieldValueText parameters

Name	Type	Description
index	ULONG	Field index

The function returns a BSTR.

NOTE As opposed to the field value, the field value text includes maps enums fields to their associated enum entry name.

8.7 QXDM automation instances

NOTE The default behavior is for QXDM to run as a single-instance automation server (one QXDM process supports all automation clients). To run as a multi-instance automation server, QXDM must be configured using the following command line syntax:

```
C:\<Installation Path> QXDM /MultiInstance
```

Note that the connection model assumes a single-instance QXDM, and well-behaved DMs turn off asynchronous traffic (Logs, Messages, and Events) before disconnecting, e.g., when switching COM ports. Automation scripts must take this into account and reconfigure settings when running multi-instance sessions.

Use the following command line syntax to restore QXDM as a single-instance automation server:

```
C:\<Installation Path> QXDM /SingleInstance
```

8.7.1 Launching QXDM from a remote computer

WARNING Making changes to a COM application using DCOM Configuration Manager affects who can access, launch, and configure the application.

It is possible to launch QXDM on a remote computer using distributed COM (DCOM). To do so, however, requires changes in the default DCOM settings on the host computer where QXDM will be running. Use the DCOMCNFG.EXE tool to give the client application permission to run QXDM:

1. From the Start → Run menu, type **`dcomcnfg`**.
2. Find QXDM from the applications list, as shown in Figure 8-1. If you cannot find it, then QXDM has not been installed properly.

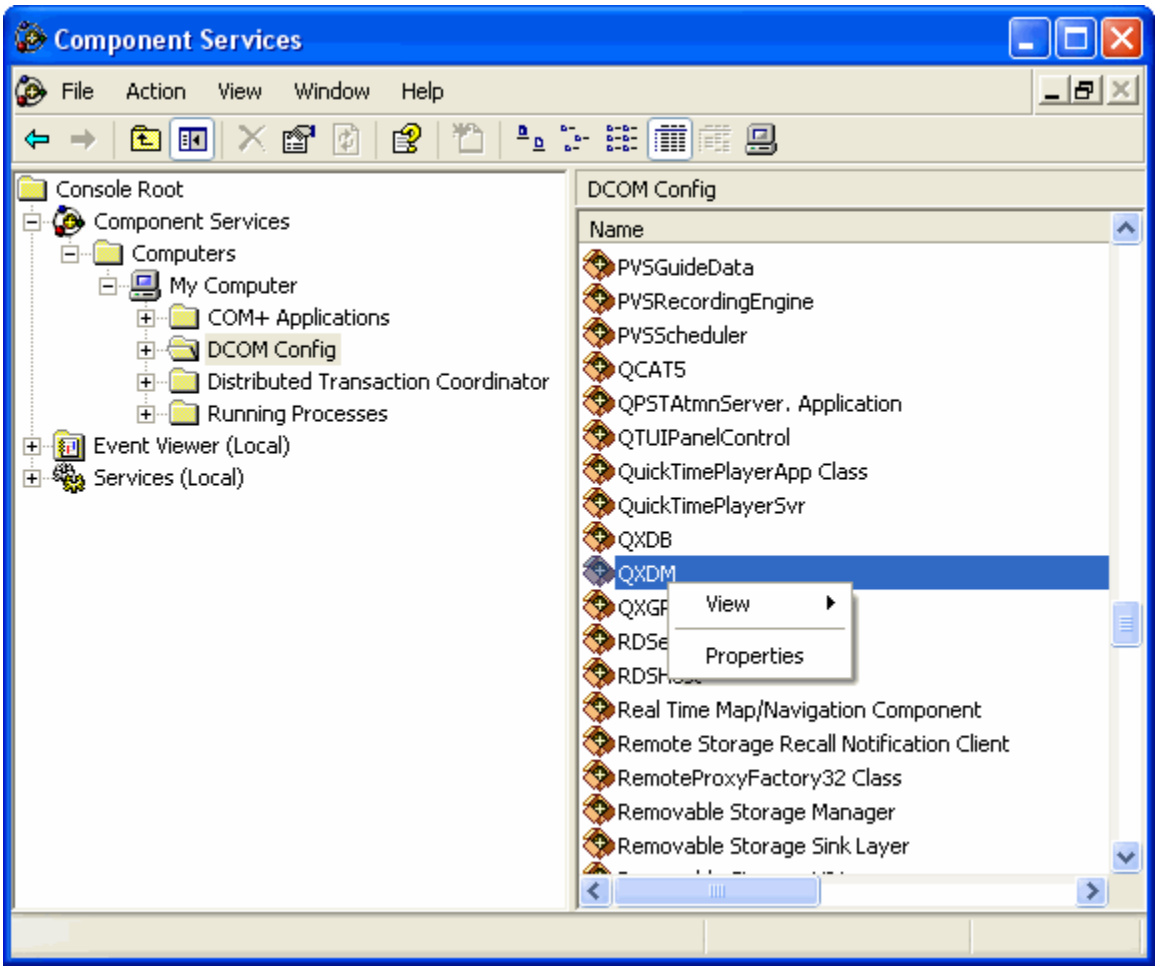

Figure 8-1 Distributed COM configuration properties (Windows XP)

3. Click PROPERTIES.

   The QXDM Properties dialog appears.

4. Click the Location tab and check the box that configures QXDM to run on a local machine, as shown in Figure 8-2.

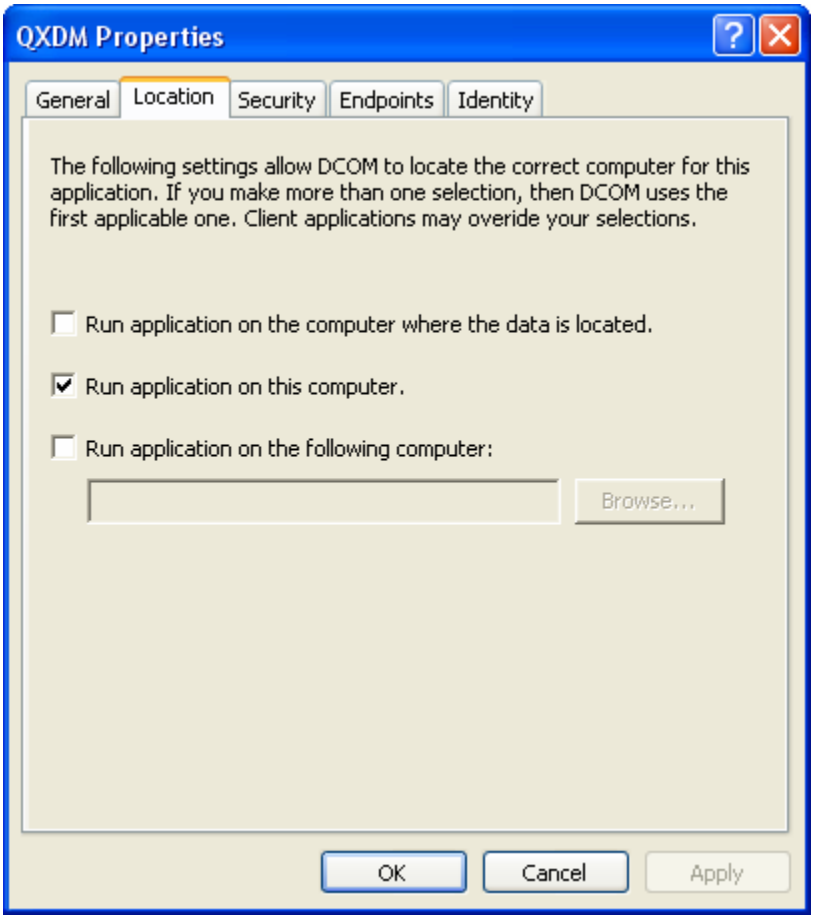

Figure 8-2 QXDM location configuration

5. Click the Security tab and select the “Use custom access permissions” radio button, as shown in Figure 8-3.

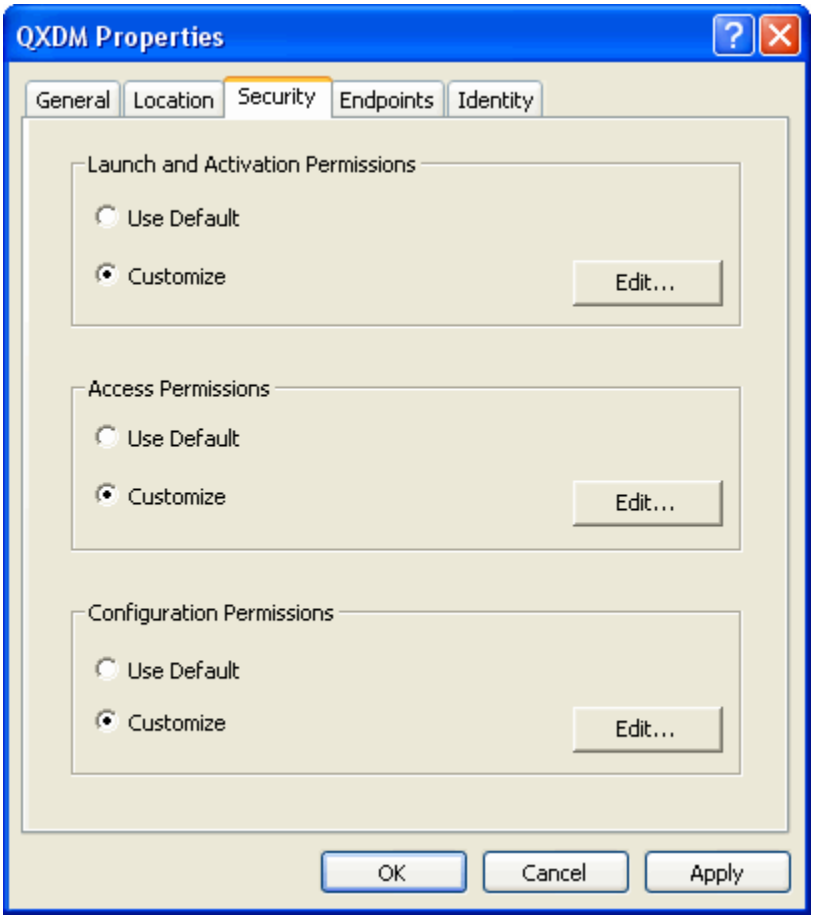

Figure 8-3 QXDM security configuration

6. Grant access, launch, and configuration permissions for trusted domains by clicking the EDIT button associated with each permission control.

NOTE Specifying who can access an application may have the side effect of restricting others from running the application (even locally) on the host computer. When customizing permissions, be sure to allow all users that need to run QXDM access. The recommended approach is to add a trusted domain of users, e.g., Domain Users. *Do not add single user accounts unless you know what you are doing.*

7. Click OK when you are through with the Securities tab.

8. Click the Identity tab and click THE INTERACTIVE USER radio button. This allows QXDM to run under an interactive account (see Figure 8-4).

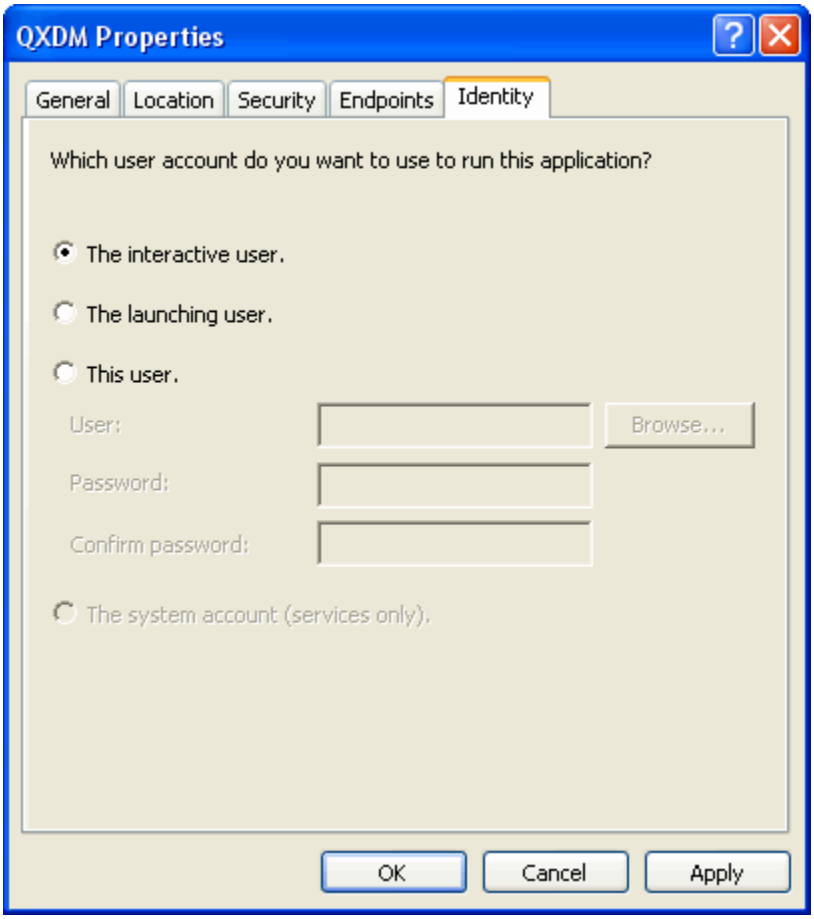

Figure 8-4 QXDM Identity configuration

9. Click OK when you are through with the Identity tab.

With all other settings as default values, you are now ready to remote launch QXDM.

NOTE Be aware that if you do not specify a server name in the code when you load the QXDM object, OLE will try to load a QXDM object from the default machine specified in the DCOM setting on the local machine.

9 Runtime Parsing DLLs

QXDM provides an interface for users to write their own parsers to parse incoming data from the phone at runtime. Microsoft Visual Studio C++ project templates are included with QXDM installations for this purpose. Two versions are provided for compatibility with VC++ 6.0 and VC++ .NET 2003 Integrated Development Environments. A readme.txt instruction file is included in each development folder located where QXDM is installed. Refer to this file for instructions on setting up and using the template projects to build custom runtime parser DLLs.

C:\Program Files\Qualcomm\QXDM\ParsingDLLWizard\Vc2005
C:\Program Files\Qualcomm\QXDM\ParsingDLLWizard\Vc2003
C:\Program Files\Qualcomm\QXDM\ParsingDLLWizard\Vc6

10 Frequently Asked Questions

Q What do I do when QXDM reports that it cannot load the Parser.dll? The error display is: “The specified module could not be found.”

A The DLL depends on another DLL that cannot be found on the target machine. You need to use a tool such as Depends or Dumpbin to better understand the dependencies of your DLL before trying to run it on other machines.

Q What do I do if the phone is not sending the logs I requested?

A Ensure that the configuration of the registering view has the log selected. Verify that QXDM has registered the logs using View Registrations (File → View Registrations). If the log is checked, then contact the responsible phone technology or build person for help in determining why the phone is not sending the log.

Q Why doesn’t the Log View display appear when I select it from the View combo-box or press <Alt + L>?

A The Log View will only appear if QXDM is connected to a phone.

Q When I start QXDM, why do I get DB Merge warnings?

A When merging user databases, QXDM reports any conflicts or errors. Refer to [Q6] for help on editing user databases.

Q The Load Items and Replay Items menu options are disabled in the File menu. How do I load an existing Item Store Format (.ISF) file?

A You cannot replay or load an Item Store while connected to a phone. In the Options menu, select Communications and make sure that the Target Port is disconnected.

Q How do I show or hide certain columns in the Item view or Filtered view?

A Right-click the view to display the context menu and then click the APPEARANCE option. Check or uncheck the box beside the column names to show or hide them.

Q How do I save my QXDM session into an Item Store Format (.ISF) file for loading or replaying at a later time?

A In the File menu, select Save Items to immediately save the current session. Alternatively, check Enable Query For ISF Save (Figure 3-9) to cause the session to be saved. When checked, QXDM will prompt for a path in which to save the .ISF file, upon exiting.

Q How do I save my configuration of open displays and registered items?

A When QXDM is open, it will open the same displays and register interest in the same items that were in the previous QXDM session. If you have multiple configurations that you would like to preserve, use the Save Configurations and Load Configurations options in the File menu to save and load QXDM configurations (see Section 3.1.4).

Q Is there any way to exclude displays from the View Bar and View menu?

A Yes, create a file named UserDisplayExclusions.txt in the QXDM database folder (typically located at C:\Documents and Settings\All Users\Documents\Qualcomm\QXDM\Database). Add to the contents of the file the name of each QXDM display (one per line) that is to be excluded from the View Bar and View menu and then restart QXDM.

1

Q Is there any way to customize the foreground and background colors used by QXDM?

A Yes, create a file named UserItemColor.txt in the QXDM database folder. Edit the content of the file according to the following rules:

- There is one line per entry.
- Fields are separated by the '^' character.
- The first field is in quotes and represents the item key.
- The second field is the red component of the color (0 to 255).
- The third field is the green component of the color (0 to 255).
- The fourth and final field is the blue component of the color (0 to 255).

The item key is a comma-delimited string that must be in the format described in Table 10-1.

Table 10-1 User color item keys

Value	Description
0	Malformed DIAG entity
1,X	DIAG response (minus following), X = command code
2,X	DIAG request (minus following), X = command code
3	GPS information
4,X	Event, X = event ID
5,X	Log, X = log code
6,X,Y	Message, X = SSID, Y = level
7,X	Generic string, X = string type
8,X,Y*,Z*	OTA log, X = log code, Y/Z vary (refer to Section 8.4.12)
9,X,Y	Subsystem dispatch response, X = SSID, Y = command code
10,X,Y	Subsystem dispatch request, X = SSID, Y = command code
0xE0000000	Background color for item list views

In nearly all cases, a value of 0xFFFFFFFF for X or Y is equal to the wildcard identifier. The precedence is for exact matches followed by wildcard matches (left to right). The exceptions to this are OTA logs where only the log code is allowed to be a wildcard and subsystem dispatch request/responses where only the subsystem command code is allowed to be a wildcard.

Example

```
"6,0xFFFFFFFF,4"^255^9^9
```

2