iSeries

Dynamic Screen Manager APIs

Version 5 Release 3

iSeries

Dynamic Screen Manager APIs

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 307.

Sixth Edition (August 2005)

This edition applies to version 5, release 3, modification 0 of Operating System/400 (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

© Copyright International Business Machines Corporation 1998, 2005. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Dynamic Screen Manager APIs 1
APIs 1
Low-Level Screen I/O Services APIs 1
Screen Manipulation and Query APIs 2
Change Low-Level Environment (QsnChgEnv) API . 3
Authorities and Locks 4
Required Parameter Group 4
Omissible Parameter Group 4
Returned Value 4
Error Messages 4
Clear Field Table (QsnClrFldTbl) API 5
Authorities and Locks 5
Omissible Parameter Group 5
Returned Value 6
Error Messages 6
Clear Screen (QsnClrScr) API 7
Authorities and Locks 7
Restrictions 7
Omissible Parameter Group 7
Returned Value 8
Error Messages 8
Create Low-Level Environment (QsnCrtEnv) API . . 9
Authorities and Locks 9
Required Parameter Group 9
Omissible Parameter Group 10
Returned Value 10
Format of the Low-Level Environment Description 10
Format of the Low-Level User Environment Extension Information 11
Field Descriptions 11
Exit Routine Error Handling 14
Change Low-Level Environment Exit Routine . . 14
Delete Low-Level Environment Exit Routine . . 14
Error Messages 15
Delete Low-Level Environment (QsnDltEnv) API . . 15
Authorities and Locks 15
Required Parameter 15
Omissible Parameter 16
Returned Value 16
Error Messages 16
Initialize Low-Level Environment Description (QsnInzEnvD) API 17
Authorities and Locks 17
Required Parameter Group 17
Omissible Parameter 18
Returned Value 18
Error Messages 18
Query 5250 (QsnQry5250) API 18
Authorities and Locks 19
Restrictions 19
Required Parameter Group 19
Omissible Parameter 19
Returned Value 19
Format of the Query Data 19
Field Descriptions 20
Error Messages 25
Query Color Support (QsnQryColorSup) API . . . 26
Authorities and Locks 26
Omissible Parameter Group 26
Returned Value 27
Error Messages 27
Query Display Mode Support (QsnQryModSup) API 27
Authorities and Locks 28
Required Parameter 28
Omissible Parameter Group 28
Returned Value 28
Error Messages 28
Restore Screen (QsnRstScr) API 29
Authorities and Locks 29
Restrictions 29
Required Parameter 29
Omissible Parameter Group 30
Returned Value 30
Error Messages 30
Retrieve Display Mode (QsnRtvMod) API 31
Authorities and Locks 31
Omissible Parameter Group 31
Returned Value 31
Error Messages 32
Retrieve Low-Level Environment Description (QsnRtvEnvD) API 32
Authorities and Locks 32
Required Parameter Group 33
Omissible Parameter Group 33
Returned Value 33
Format of the Data Returned 33
Error Messages 33
Retrieve Low-Level Environment User Data (QsnRtvEnvDta) API 34
Authorities and Locks 34
Required Parameter 34
Omissible Parameter Group 34
Returned Value 35
Error Messages 35
Retrieve Low-Level Environment Window Mode (QsnRtvEnvWinMod) API 36
Authorities and Locks 36
Required Parameter Group 36
Omissible Parameter Group 37
Returned Value 37
Format of the Data Returned 37
Field Descriptions 37
Error Messages 37
Retrieve Screen Dimensions (QsnRtvScrDim) API. . 38
Authorities and Locks 38
Omissible Parameter Group 38
Returned Value 39
Error Messages 39
Roll Down (QsnRollDown) API. 40
Restrictions 40
Authorities and Locks 41

© Copyright IBM Corp. 1998, 2005

Required Parameter Group 41
Omissible Parameter Group 41
Returned Value 41
Error Messages 41
Roll Up (QsnRollUp) API 42
Usage Notes 42
Authorities and Locks 42
Required Parameter Group 43
Omissible Parameter Group 43
Returned Value 43
Error Messages 43
Save Screen (QsnSavScr) API 44
Authorities and Locks 44
Restrictions 45
Omissible Parameter Group 45
Returned Value 45
Error Messages 45
Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API 46
Authorities and Locks 48
Required Parameter 48
Omissible Parameter Group 48
Returned Value 49
Format of the Window Mode Description . . . 49
Field Descriptions 49
Error Messages 50
Buffer Manipulation and Query APIs 50
Clear Buffer (QsnClrBuf) API 51
Authorities and Locks 52
Required Parameter 52
Omissible Parameter 52
Returned Value 52
Error Messages 52
Copy Buffer (QsnCpyBuf) API 53
Authorities and Locks 53
Required Parameter Group 53
Omissible Parameter 54
Returned Value 54
Error Messages 54
Create Command Buffer (QsnCrtCmdBuf) API . . 55
Authorities and Locks 55
Required Parameter 55
Omissible Parameter Group 55
Returned Value 56
Error Messages 56
Create Input Buffer (QsnCrtInpBuf) API 57
Authorities and Locks 57
Required Parameter 57
Omissible Parameter Group 57
Returned Value 58
Error Messages 58
Delete Buffer (QsnDltBuf) API 59
Authorities and Locks 59
Required Parameter 59
Omissible Parameter 59
Returned Value 59
Error Messages 60
Put Command Buffer (QsnPutBuf) API 60
Authorities and Locks 60
Required Parameter 60
Omissible Parameter Group 61
Returned Value 61
Error Messages 61
Put Command Buffer and Perform Get (QsnPutGetBuf) API 62
Authorities and Locks 62
Required Parameter Group 62
Omissible Parameter Group 62
Returned Value 63
Error Messages 63
Retrieve AID Code on Read (QsnRtvReadAID) API 64
Authorities and Locks 64
Required Parameter 64
Omissible Parameter Group 64
Returned Value 65
Error Messages 65
Retrieve Available Data (QsnRtvAvailData) API . . 65
Authorities and Locks 66
Required Parameter Group 66
Omissible Parameter Group 66
Returned Value 66
Error Messages 66
Retrieve Buffer Data Length (QsnRtvBufLen) API. . 67
Authorities and Locks 67
Required Parameter 67
Omissible Parameter Group 67
Returned Value 68
Error Messages 68
Retrieve Buffer Size (QsnRtvBufSiz) API 68
Authorities and Locks 69
Required Parameter 69
Omissible Parameter Group 69
Returned Value 69
Error Messages 69
Retrieve Cursor Address on Read (QsnRtvReadAdr) API 70
Authorities and Locks 70
Required Parameter 70
Omissible Parameter Group 71
Returned Value 71
Error Messages 71
Retrieve Field Information (QsnRtvFldInf) API . . 72
Authorities and Locks 72
Required Parameter Group 73
Omissible Parameter Group 73
Returned Value 73
Format of the Query Input Field Result 73
Field Descriptions 74
Error Messages 74
Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API 75
Authorities and Locks 75
Required Parameter 75
Omissible Parameter Group 75
Returned Value 75
Error Messages 76
Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API 76
Authorities and Locks 77
Required Parameter 77
Omissible Parameter Group 77
Returned Value 77

Error Messages 77
Retrieve Number of Bytes Read from Screen (QsnRtvReadLen) API 78
Authorities and Locks 78
Required Parameter 78
Omissible Parameter Group 78
Returned Value 79
Error Messages 79
Retrieve Number of Fields Read (QsnRtvFldCnt) API 79
Authorities and Locks 80
Required Parameter 80
Omissible Parameter Group 80
Returned Value 80
Error Messages 80
Retrieve Pointer to Data in Input Buffer (QsnRtvDta) API 81
Authorities and Locks 81
Required Parameter 81
Omissible Parameter Group 81
Returned Value 82
Error Messages 82
Retrieve Pointer to Field Data (QsnRtvFldDta) API 82
Authorities and Locks 83
Required Parameter 83
Omissible Parameter Group 83
Returned Value 83
Error Messages 83
Retrieve Read Information (QsnRtvReadInf) API . . 84
Authorities and Locks 84
Required Parameter Group 84
Omissible Parameter Group 85
Returned Value 85
Format of the Query Result 85
Field Descriptions 85
Error Messages 86
Screen Input APIs 87
Get AID (QsnGetAID) API 88
Authorities and Locks 88
Omissible Parameter Group 88
Returned Value 88
Error Messages 89
Get Cursor Address (QsnGetCsrAdr) API 89
Authorities and Locks 90
Restrictions 90
Omissible Parameter Group 90
Returned Value 90
Error Messages 90
Get Cursor Address with AID (QsnGetCsrAdrAID) API 91
Authorities and Locks 91
Omissible Parameter Group 91
Returned Value 92
Error Messages 92
Put Input Command (QsnPutInpCmd) API 93
Authorities and Locks 94
Required Parameter 94
Omissible Parameter Group 94
Returned Value 95
Error Messages 95
Read from Invited Device (QsnReadInvited) API . . 96
Authorities and Locks 96
Restrictions 96
Required Parameter Group 96
Omissible Parameter Group 97
Returned Value 97
Error Messages 97
Read Immediate (QsnReadImm) API 98
Authorities and Locks 99
Restrictions 99
Omissible Parameter Group 99
Returned Value 99
Error Messages 99
Read Input Fields (QsnReadInp) API 100
Restrictions 101
Authorities and Locks 102
Required Parameter Group 102
Omissible Parameter Group 102
Returned Value 102
Error Messages 103
Read Modified Alternate (QsnReadMDTAlt) API 104
Authorities and Locks 104
Restrictions 104
Required Parameter Group 105
Omissible Parameter Group 105
Returned Value 105
Error Messages 106
Read Modified Fields (QsnReadMDT) API. . . . 107
Restrictions 108
Authorities and Locks 108
Required Parameter Group 109
Omissible Parameter Group 109
Returned Value 109
Error Messages 109
Read Modified Immediate Alternate (QsnReadMDTImmAlt) API 110
Restrictions 111
Authorities and Locks 111
Omissible Parameter Group 111
Returned Value 111
Error Messages 111
Read Screen (QsnReadScr) API 112
Authorities and Locks 113
Restrictions 113
Omissible Parameter Group 113
Returned Value 113
Error Messages 114
Screen Output APIs 114
Delete Field ID Definition (QsnDltFldId) API . . . 115
Authorities and Locks 115
Required Parameter 115
Omissible Parameter 115
Returned Value 116
Error Messages 116
Generate a Beep (QsnBeep) API 116
Authorities and Locks 116
Restrictions 117
Omissible Parameter Group 117
Returned Value 117
Error Messages 117
Insert Cursor (QsnInsCsr) API 118
Authorities and Locks 118

Restrictions 118
Omissible Parameter Group 119
Returned Value 119
Error Messages 120
Pad between Two Screen Addresses (QsnWrtPadAdr) API 121
Authorities and Locks 121
Restrictions 122
Required Parameter Group 122
Omissible Parameter Group 122
Returned Value 123
Error Messages 123
Pad for N Positions (QsnWrtPad) API 124
Authorities and Locks 125
Restrictions 125
Required Parameter Group 125
Omissible Parameter Group 125
Returned Value 126
Error Messages 126
Put Output Command (QsnPutOutCmd) API. . . 127
Authorities and Locks 127
Required Parameter 127
Omissible Parameter Group 128
Returned Value 128
Error Messages 128
Set Cursor Address (QsnSetCsrAdr) API 129
Authorities and Locks 130
Restrictions 130
Omissible Parameter Group 130
Returned Value 131
Error Messages 131
Set Error State (QsnSetErr) API 132
Authorities and Locks 133
Omissible Parameter Group 133
Returned Value 134
Error Messages 134
Set Field (QsnSetFld) API 136
Authorities and Locks 137
Restrictions 137
Omissible Parameter Group 137
Returned Value 139
Format of the Field Format Word 140
Format of the Field Control Word 143
Valid Field Control Word and Device Capability Combinations 144
Error Messages 144
Set Field with CCSID (QsnSetFldCC) API 146
Restrictions 147
Authorities and Locks 147
Omissible Parameter Group 147
Return Value 150
Format of the Field Control Word 150
Valid Field Control Word and Device Capability Combinations 151
Error Messages 151
Set Output Address (QsnSetOutAdr) API 152
Authorities and Locks 152
Omissible Parameter Group 152
Returned Value 153
Error Messages 153
Write Data (QsnWrtDta) API 155
Restrictions 156
Authorities and Locks 156
Required Parameter Group 156
Omissible Parameter Group 156
Returned Value 158
Error Messages 158
Write Data with CCSID (QsnWrtDtaCC) API . . . 159
Restrictions 160
Authorities and Locks 160
Required Parameter Group 160
Omissible Parameter Group 160
Returned Value 162
Error Messages 162
Write Structured Field Major (QsnWrtSFMaj) API 163
Authorities and Locks 164
Restrictions 164
Required Parameter Group 164
Omissible Parameter Group 164
Returned Value 165
Error Messages 165
Write Structured Field Minor (QsnWrtSFMin) API 166
Authorities and Locks 167
Restrictions 167
Required Parameter Group 167
Omissible Parameter Group 167
Returned Value 167
Error Messages 167
Write to Display (QsnWTD) API 169
Authorities and Locks 170
Required Parameter Group 170
Omissible Parameter Group 170
Returned Value 171
Error Messages 171
Write Transparent Data (QsnWrtTDta) API . . . 172
Authorities and Locks 173
Restrictions 173
Required Parameter Group 173
Omissible Parameter Group 173
Returned Value 174
Error Messages 175
Window Services APIs 175
Window Manipulation and Query APIs 175
Change Window (QsnChgWin) API 176
Authorities and Locks 176
Required Parameter Group 177
Omissible Parameter 177
Returned Value 177
Error Messages 177
Create a Window (QsnCrtWin) API 178
Authorities and Locks 178
Required Parameter Group 179
Omissible Parameter Group 179
Returned Value 180
Restrictions 180
Format of the Window Description 180
Field Descriptions 181
Format of the Window User Extension Information 185
Field Descriptions 185
Window Exit Routines 185
Exit Routine Error Handling 185

Change Window Exit Routine 185
Delete Window Exit Routine 186
Change Window Coordinates Exit Routine . . 186
Draw Window Exit Routine 188
Current Window Exit Routine 188
Error Messages 188
Initialize Window Description (QsnInzWinD) API 189
Authorities and Locks 190
Required Parameter Group 190
Omissible Parameter Group 190
Returned Value 190
Error Messages 190
Move Window (QsnMovWin) API 191
Authorities and Locks 191
Required Parameter Group 191
Omissible Parameter 192
Returned Value 192
Error Messages 192
Move Window by User (QsnMovWinUsr) API . . 193
Authorities and Locks 193
Required Parameter 193
Omissible Parameter 193
Returned Value 193
Error Messages 194
Resize Window (QsnRszWin) API 194
Authorities and Locks 195
Required Parameter Group 195
Omissible Parameter 195
Returned Value 195
Error Messages 195
Resize Window by User (QsnRszWinUsr) API . . 196
Authorities and Locks 196
Required Parameter 196
Omissible Parameter 197
Returned Value 197
Error Messages 197
Retrieve Window Data (QsnRtvWinDta) API . . . 198
Authorities and Locks 198
Required Parameter 198
Omissible Parameter Group 198
Returned Value 199
Error Messages 199
Retrieve Window Description (QsnRtvWinD) API 199
Authorities and Locks 200
Required Parameter Group 200
Omissible Parameter 200
Returned Value 200
Format of the Window Description Returned 200
Field Descriptions 200
Error Messages 201
Set Window Services Attributes (QsnSetWinAtr) API. 201
Authorities and Locks 201
Required Parameter Group 201
Omissible Parameter 202
Returned Value 202
Format of the Window Services Attribute Description 202
Field Descriptions 202
Error Messages 203
Window I/O APIs 203
Clear Window (QsnClrWin) API 204
Authorities and Locks 204
Required Parameter 204
Omissible Parameter 204
Returned Value 204
Error Messages 205
Clear Window Message (QsnClrWinMsg) API . . 205
Authorities and Locks 205
Required Parameter 205
Omissible Parameter 206
Returned Value 206
Error Messages 206
Display Window (QsnDspWin) API 207
Authorities and Locks 207
Required Parameter 207
Omissible Parameter 207
Returned Value 207
Error Messages 208
Put Window Message (QsnPutWinMsg) API . . . 209
Authorities and Locks 210
Required Parameter 210
Omissible Parameter Group 210
Returned Value 211
Error Messages 212
Window Manager Services APIs 212
End a Window (QsnEndWin) API 213
Authorities and Locks 213
Required Parameter 213
Omissible Parameter Group 213
Returned Value 214
Error Messages 214
Retrieve Current Window (QsnRtvCurWin) API 214
Authorities and Locks 215
Omissible Parameter Group 215
Returned Value 215
Error Messages 215
Set Current Window (QsnSetCurWin) API. 216
Authorities and Locks 216
Required Parameter 216
Omissible Parameter 216
Returned Value 216
Error Messages 217
Start a Window (QsnStrWin) API 217
Authorities and Locks 217
Required Parameter 218
Omissible Parameter Group 218
Returned Value 218
Error Messages 218
Performance Considerations 218
Creating/Manipulating Windows Example . . 219
Session Services APIs 221
Session Manipulation and Query APIs 222
Change Session (QsnChgSsn) API 223
Authorities and Locks 223
Required Parameter Group 223
Omissible Parameter 224
Returned Value 224
Error Messages 224
Clear Scroller (QsnClrScl) API 225
Authorities and Locks 225
Required Parameter 225

Omissible Parameter Group 225
Returned Value 226
Error Messages 226
Create a Session (QsnCrtSsn) API. 227
Authorities and Locks 227
Required Parameter Group 228
Omissible Parameter Group 228
Returned Value 229
Format of the Session Description 229
Field Descriptions 230
Format of the Session User Extension Data . . 233
Field Descriptions 233
Session Exit Routines 233
Change Session Exit Routine 234
Change Session Exit Routine Parameter . . . 234
Delete Session Exit Routine 234
Delete Session Exit Routine Parameter 234
Change Session Coordinates Exit Routine . . . 234
Change Session Coordinates Exit Routine Parameters 235
Exit Routine Error Handling 236
Draw Session Exit Routine 236
Draw Session Exit Routine Parameters 236
Current Session Exit Routine 236
Current Session Exit Routine Parameter . . . 237
Error Messages 237
Display Scroller Bottom (QsnDspSclB) API . . . 238
Authorities and Locks 238
Required Parameter 238
Omissible Parameter 238
Returned Value 238
Error Messages 239
Display Scroller top (QsnDspSclT) API 239
Authorities and Locks 239
Required Parameter 239
Omissible Parameter 240
Returned Value 240
Error Messages 240
Initialize Session Description (QsnInzSsnD) API 241
Authorities and Locks 241
Required Parameter Group 241
Omissible Parameter 241
Returned Value 242
Error Messages 242
Query If Scroller in Line Wrap Mode (QsnQrySclWrp) API 242
Authorities and Locks 242
Required Parameter 243
Omissible Parameter Group 243
Returned Value 243
Error Messages 243
Retrieve Number of Columns to Shift Scroller (QsnRtvSclNumShf) API. 244
Authorities and Locks 244
Required Parameter 244
Omissible Parameter Group 244
Returned Value 245
Error Messages 245
Retrieve Number of Rows to Roll Scroller (QsnRtvSclNumRoll) API 245
Authorities and Locks 245
Required Parameter 246
Omissible Parameter Group 246
Returned Value 246
Error Messages 246
Retrieve Session Data (QsnRtvSsnDta) API . . . 247
Authorities and Locks 247
Required Parameter 247
Omissible Parameter Group 247
Returned Value 248
Error Messages 248
Retrieve Session Description (QsnRtvSsnD) API 248
Authorities and Locks 249
Required Parameter Group 249
Omissible Parameter 249
Returned Value 249
Format of the Session Description Returned . . 249
Error Messages 250
Roll Scroller Down (QsnRollSclDown) API . . . 250
Authorities and Locks 250
Required Parameter 251
Omissible Parameter Group 251
Returned Value 251
Error Messages 251
Roll Scroller Up (QsnRollSclUp) API 252
Authorities and Locks 252
Required Parameter 252
Omissible Parameter Group 252
Returned Value 253
Error Messages 253
Shift Scroller left (QsnShfSclL) API 253
Restrictions 254
Authorities and Locks 254
Required Parameter 254
Omissible Parameter Group 254
Returned Value 254
Error Messages 254
Shift Scroller Right (QsnShfSclR) API 255
Restrictions 255
Authorities and Locks 255
Required Parameter 255
Omissible Parameter Group 255
Returned Value 256
Error Messages 256
Toggle Line Wrap/Truncate Mode (QsnTglSclWrp) API. 257
Authorities and Locks 257
Required Parameter 257
Omissible Parameter Group 257
Returned Value 258
Error Messages 258
Session I/O APIs 258
Backspace on Scroller Line (QsnSclBS) API . . . 259
Authorities and Locks 259
Required Parameter 259
Omissible Parameter 259
Returned Value 260
Error Messages 260
Go to Next Tab Position in Scroller Line (QsnSclTab) API 260
Authorities and Locks 260
Required Parameter 261

Omissible Parameter 261
Returned Value 261
Error Messages 261
Go to Start of Current Scroller Line (QsnSclCR) API 262
Authorities and Locks 262
Required Parameter 262
Omissible Parameter 262
Returned Value 262
Error Messages 263
Go to Start of Next Scroller Line (QsnSclNL) API 263
Authorities and Locks 263
Required Parameter 263
Omissible Parameter 264
Returned Value 264
Error Messages 264
Print Scroller Data (QsnPrtScl) API 264
Restrictions 265
Authorities and Locks 265
Required Parameter 265
Omissible Parameter 265
Returned Value 265
Error Messages 265
Read Data from Session (QsnReadSsnDta) API . . 266
Restrictions 266
Authorities and Locks 267
Required Parameter Group 267
Omissible Parameter Group 267
Returned Value 267
Error Messages 267
Read Data from Session with CCSID (QsnReadSsnDtaCC) API 268
Restrictions 269
Authorities and Locks 269
Required Parameter Group 269
Omissible Parameter Group 269
Returned Value 270
Error Messages 270
Retrieve Session Line to Input Line (QsnRtvSsnLin) API 270
Restrictions 271
Authorities and Locks 271
Required Parameter 271
Omissible Parameter 271
Returned Value 271
Error Messages 271
Start New Scroller Line at Current Position (QsnSclLF) API 272
Authorities and Locks 272
Required Parameter 272
Omissible Parameter 272
Returned Value 272
Error Messages 273
Start New Scroller Page (QsnSclFF) API 273
Authorities and Locks 273
Required Parameter 273
Omissible Parameter 274
Returned Value 274
Error Messages 274
Write Characters to Scroller (QsnWrtSclChr) API 275
Authorities and Locks 275
Required Parameter Group 275
Omissible Parameter 276
Returned Value 276
Error Messages 276
Write Characters to Scroller with CCSID (QsnWrtSclChrCC) API 277
Restrictions 277
Authorities and Locks 277
Required Parameter Group 277
Omissible Parameter 278
Error Messages 278
Write Line to Scroller (QsnWrtSclLin) API 279
Authorities and Locks 279
Required Parameter Group 279
Omissible Parameter 280
Returned Value 280
Error Messages 280
Write Line to Scroller with CCSID (QsnWrtSclLinCC) API 281
Restrictions 281
Authorities and Locks 281
Required Parameter Group 282
Omissible Parameter 282
Error Messages 282
Concepts 283
Using Dynamic Screen Manager APIs 283
Data Structures for DSM APIs 283
Omitting Parameters with Associated Lengths 283
Low-Level Services Examples 283
Low-Level Services Example—1 283
Low-Level Services Example—2 284
Low-Level Services Example—3 286
Low-Level Services Example—4 289
Using Low-Level Screen I/O Services APIs . . . 291
DSM Error Handling 291
Device Support 291
Operating Environments 291
Direct and Indirect Operations 291
DBCS Considerations 292
Performance Considerations 292
Limitations and Restrictions 292
5250 Data Stream Details 293
AID-Generating Keys 293
Control Characters 293
Screen Attribute Characters 295
Display Address 296
Insert Cursor Address 296
Modified Data Tag (MDT) Bit 296
Resequencing 296
States and Modes 297
Dumping the 5250 Data Stream Commands . . 297
Using Window Services APIs 297
Using Session Services APIs 301
Session Details 301
Line Mode and Character Mode I/O 302
Command Key Action Routines 302
Action Routine Parameters 303
Active Position 304
EBCDIC Display Control Characters 304
DBCS Considerations 305

Appendix. Notices 307
Trademarks 308
Terms and conditions for downloading and printing publications 309
Code disclaimer information 310

Dynamic Screen Manager APIs

The Dynamic Screen Manager (DSM) APIs are a set of screen I/O interfaces that provide a dynamic way to create and manage screens for the Integrated Language Environment (ILE) high-level languages. Because the DSM interfaces are bindable, they are accessible to ILE programs only.

The DSM APIs provide an alternative to the existing way of defining screen appearance outside a program by coding in DDS or UIM, for example. Instead, programmers can use a series of calls to DSM within their programs to dynamically specify and control screen appearance for their applications. Unlike static definition methods, the DSM interfaces provide the flexibility needed for those applications requiring more dynamic screen control. The DSM support provided varies from low-level interfaces for direct screen manipulation to windowing support.

The DSM APIs fall into the following functional groups:

- "Low-Level Screen I/O Services APIs" provide a direct interface to the 5250 data stream commands. These APIs are used to query and manipulate the state of the screen; to create, query, and manipulate input and command buffers used to interact with the screen; and to define fields and write data to the screen.
- "Window Services APIs" on page 175 are used to create, delete, move, and resize windows, and to manage multiple windows during a session.
- "Session Services APIs" on page 221 provide a general scrolling interface that can be used to create, query, and manipulate sessions, and to perform input and output operations to sessions.

See "Using Dynamic Screen Manager APIs" on page 283 for additional information.

See Code disclaimer information for information pertaining to code examples.

Top | APIs by category

APIs

These are the APIs for this category.

Low-Level Screen I/O Services APIs

The low-level services are divided into the following functional groups:

- "Screen Manipulation and Query APIs" on page 2, which are used to query and manipulate the state of the screen.
- "Buffer Manipulation and Query APIs" on page 50, which are used to create, query, and manipulate input and command buffers used to interact with the screen.
- "Screen Input APIs" on page 87, which are used to read field data and other information from the screen.
- "Screen Output APIs" on page 114, which are used to define fields and write data and other information to the screen.

The low-level screen I/O services APIs provide a direct interface to the 5250 data stream commands and user-defined data streams for performing basic screen I/O operations. For detailed information about the results and interactions of these operations, refer to the following 5250 data stream documentation:

- *5250 Functions Reference*, SA21-9247-06. This book is not accessible online, but can be ordered from the IBM Publications Center

© Copyright IBM Corp. 1998, 2005

.

- *5494 Remote Control Unit Functions Reference R3.1*, SC30-3533-04. This book can be viewed online through the IBM Publications Center .

For additional information, select one of the following:

- "Low-Level Services Examples" on page 283
- "DSM Error Handling" on page 291
- "Device Support" on page 291
- "Operating Environments" on page 291
- "Direct and Indirect Operations" on page 291
- "DBCS Considerations" on page 292
- "Performance Considerations" on page 292
- "Limitations and Restrictions" on page 292
- "5250 Data Stream Details" on page 293, including
  - "5250 Data Stream Details" on page 293
  - "Control Characters" on page 293
  - "Screen Attribute Characters" on page 295
  - "Display Address" on page 296
  - "Insert Cursor Address" on page 296
  - "Modified Data Tag (MDT) Bit" on page 296
  - "Resequencing" on page 296
  - "States and Modes" on page 297
  - "Dumping the 5250 Data Stream Commands" on page 297

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Screen Manipulation and Query APIs

The screen manipulation and query APIs are used to query and manipulate the state of the screen. They are:

- "Change Low-Level Environment (QsnChgEnv) API" on page 3 (QsnChgEnv) changes the low-level environment description.
- "Clear Field Table (QsnClrFldTbl) API" on page 5 (QsnClrFldTbl) clears the format table but does not alter the display.
- "Clear Screen (QsnClrScr) API" on page 7 (QsnClrScr) clears the screen and sets the screen size.
- "Create Low-Level Environment (QsnCrtEnv) API" on page 9 (QsnCrtEnv) creates the environment for the low-level services APIs.
- "Delete Low-Level Environment (QsnDltEnv) API" on page 15 (QsnDltEnv) deletes the environment for the low-level services APIs.
- "Initialize Low-Level Environment Description (QsnInzEnvD) API" on page 17 (QsnInzEnvD) initializes the low-level environment description.
- "Query 5250 (QsnQry5250) API" on page 18 (QsnQry5250) returns the results of the 5250 Query command.
- "Query Color Support (QsnQryColorSup) API" on page 26 (QsnQryColorSup) determines if the display device supports color.

- "Query Display Mode Support (QsnQryModSup) API" on page 27 (QsnQryModSup) queries if the display device allows a given mode.
- "Restore Screen (QsnRstScr) API" on page 29 (QsnRstScr) restores the contents of a screen saved with Save Screen (QsnSavScr) API.
- "Retrieve Display Mode (QsnRtvMod) API" on page 31 (QsnRtvMod) queries the current device mode.
- "Retrieve Low-Level Environment Description (QsnRtvEnvD) API" on page 32 (QsnRtvEnvD) retrieves the low-level environment description.
- "Retrieve Low-Level Environment User Data (QsnRtvEnvDta) API" on page 34 (QsnRtvEnvDta) returns a pointer to the user data for the low-level environment.
- "Retrieve Low-Level Environment Window Mode (QsnRtvEnvWinMod) API" on page 36 (QsnRtvEnvWinMod) retrieves the low-level interface environment window mode description.
- "Retrieve Screen Dimensions (QsnRtvScrDim) API" on page 38 (QsnRtvScrDim) retrieves the screen dimensions.
- "Roll Down (QsnRollDown) API" on page 40 (QsnRollDown) rolls the screen down by the given number of rows.
- "Roll Up (QsnRollUp) API" on page 42 (QsnRollUp) rolls the screen up by the given number of rows.
- "Save Screen (QsnSavScr) API" on page 44 (QsnSavScr) saves the present display so it can be restored later.
- "Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API" on page 46 (QsnSetEnvWinMod) sets the window mode for the low-level interface environment.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Change Low-Level Environment (QsnChgEnv) API

Required Parameter Group:

1	Low-level environment description
Input	Char(*)
2	Length of low-level environment description
Input	Binary(4)

Omissible Parameter Group:

3	Low-level environment handle
Input	Binary(4)
4	Error Code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Change Low-Level Environment (QsnChgEnv) API changes the description for the given low-level environment. The Change Low-Level Environment exit routine will be called if specified on the user extension information of the Create Low-Level Environment (QsnCrtEnv) API.

Authorities and Locks

Display file authority
*USE

Display file library authority
*USE

Exit routine authority
*EXECUTE

Required Parameter Group

Low-level environment description
INPUT; CHAR(*)

The new environment description for the given environment. The format of this parameter is shown in "Format of the Low-Level Environment Description" on page 10.

Length of low-level environment description
INPUT; Binary(4)

The length of the low-level environment description parameter. The value specified must be 16, 36 or 38.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment to be modified. If this parameter is omitted or specified as 0, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.

Message ID	Error Message Text
CPFA327 E	Low level environment description value incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA344 E	The file &2 in library &3 is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Clear Field Table (QsnClrFldTbl) API

Omissible Parameter Group:

1 Command buffer handle

Input Binary(4)

2 Low-level environment handle

Input Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Clear Field Table (QsnClrFldTbl) API erases the contents of the format table without affecting the display station screen. This allows for example, all input field definitions to be erased from the screen without altering the physical appearance of the screen.

This command corresponds directly to the 5250 Clear Format Table command. See the 5250 data stream documentation for details.

Authorities and Locks

None

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

If this parameter is omitted or specified as 0, this is a direct operation and the format table is cleared immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Clear Screen (QsnClrScr) API

Omissible Parameter Group:	
1	Mode
Input	Char(1)
2	Command buffer handle
Input	Binary(4)
3	Low-level environment handle
Input	Binary(4)
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Clear Screen (QsnClrScr) API clears the screen and sets the screen size to the specified mode. This command corresponds directly to the 5250 Clear Unit or Clear Unit Alternate command, depending upon the current screen presentation size. See the 5250 data stream documentation for details.

Authorities and Locks

None

Restrictions

If this is an indirect operation, it must be the first command in the command buffer.

Omissible Parameter Group

Mode INPUT; CHAR(1)

The mode to place the screen in after the screen is cleared. If this parameter is omitted, a value of 0 is assumed.

The possible values are:

0	Indicates that the current screen size should be kept. For indirect operations where this value is specified, the subsequent clear operation will be based on the current screen size, not on whatever size the screen is when the command buffer is ultimately written out. The current display size will be determined using the QsnRtvMod interface.
3	Set screen to 24x80 mode.
4	Set screen to 27x132 mode. This value is not supported by all devices. A CPFA306 error will occur if an attempt is made to specify this value with a device that does not support it.

Command buffer handle
INPUT; BINARY(4)

If this parameter is omitted or specified as 0, this is a direct operation and the screen is cleared immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA306 E	Command not supported by current device.
CPFA321 E	Operation not first command in command buffer.
CPFA322 E	Incorrect display mode &1 specified.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Create Low-Level Environment (QsnCrtEnv) API

Required Parameter Group:	
1	Low-level environment description
Input	Char(*)
2	Length of low-level environment description
Input	Binary(4)
Omissible Parameter Group:	
3	User Extension Information
Input	Char(*)
4	Length of user extension information
Input	Binary(4)
5	Low-level environment handle
Output	Binary(4)
6	Error Code
I/O	Char(*)
Returned Value:	
	Low-level environment handle
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Create Low-Level Environment (QsnCrtEnv) API creates an operating environment for low-level interface routines.

Authorities and Locks

Display file authority
*USE

Display file library authority
*USE

Exit routine authority
*EXECUTE

Required Parameter Group

Low-level environment description
INPUT; CHAR(*)

The environment description for the low-level interfaces. The format of this parameter is shown in "Format of the Low-Level Environment Description" on page 10.

Length of low-level environment description
INPUT; Binary(4)

The length of the low-level environment description parameter. The value specified must be 16, 36, or 38.

Omissible Parameter Group

User extension information
INPUT; CHAR(*)

The user extension information is used to associate data and exit routines with the low-level environment. This essentially enables the object-oriented programming concept of inheritance, allowing the low-level environment to be extended in a natural way. The user extension information cannot be changed once the environment has been created. The format of this parameter is shown in "Format of the Low-Level User Environment Extension Information" on page 11.

User extension information length
INPUT; BINARY(4)

The length of the user extension information parameter. If this parameter is specified with a zero value, the user extension information parameter is ignored. If a non-zero value is specified, it must be 48 and the user extension parameter is required.

Low-level environment handle
OUTPUT; BINARY(4)

The low-level environment that is created as a result of this operation. This handle can be used across activation groups if the activation group in which the handle was created is still active.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Low-level environment handle
OUTPUT; BINARY(4)

The value for parameter 5 is returned. If an error occurs during processing, returns -1.

Format of the Low-Level Environment Description

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Color support
1	1	CHAR(1)	Character conversion
2	2	CHAR(1)	CDRA conversions to 0x3F
3	3	CHAR(1)	DBCS support
4	4	CHAR(1)	Coexistence
5	5	CHAR(1)	Alternative help key support
6	6	CHAR(10)	Target device
16	10	CHAR(20)	Display file
36	24	CHAR(1)	Invite active

Offset		Type	Field
Dec	Hex		
37	25	CHAR(1)	Prevent override

Format of the Low-Level User Environment Extension Information

Offset		Type	Field
Dec	Hex		
0	0	PTR(SPP)	User data associated with the environment
16	10	PTR(PP)	Exit routine to call when the low-level environment is changed
32	20	PTR(PP)	Exit routine to call when the low-level environment is deleted

Field Descriptions

In the following descriptions, specifying the value *Same* indicates the current value when using the Change Low-Level Environment (QsnChgEnv) API. The default value refers to the value set by the Initialize Low-Level Environment Description (QsnInzEnvD) API.

Alternative help key support. Specifies if the alternative help key is used. The default is no alternative help key support.

The possible values for this field are:

0 Same: Keep the current setting.

1 None: No alternative help key support is used.

QSN_F1 through QSN_F24 The specified key is the alternative help key. See Table 1 for the values that correspond to these mnemonics. When this key is pressed, the AID code for the Help key will be returned.

CDRA conversions to X'3F'. When CDRA conversion takes place, all characters not supported in the target CCSID are converted to X'3F'. Sending data containing X'3F' to the display causes adverse effects.

This field specifies whether the DSM low-level routines are to check for X'3F' in the data to be displayed and perform any conversions if necessary. Conversion will be performed for both direct and indirect operations on data output through the QsnWrtDta and QsnWrtTDta APIs, and input through the QsnReadInp, QsnReadMDT, QsnReadMDTAlt, QsnReadMDTAlt, QsnReadImm, and QsnReadMDTImmAlt APIs.

The default is convert if character conversion (see below) is specified as convert. Otherwise, the default is standard. (See "Limitations and Restrictions" on page 292 for further details.)

The possible values for this field are:

0 Same: Keep the current setting.

1 Standard: Always display data as standard data with no replacement.

2	Convert: Always check for X'3F' in data and convert to X'1F' before displaying the data and convert X'1F' in incoming data to X'3F'.

Character conversion. This field specifies whether CDRA conversion takes place on the data when the job CCSID does not match that of the display device. Conversion will be performed for both direct and indirect operations on data output through the QsnWrtDta and QsnWrtTDta APIs, and input through the QsnReadInp, QsnReadMDT, QsnReadMDTAlt, QsnReadMDTAlt, QsnReadImm, and QsnReadMDTImmAlt APIs.

The CCSID for the display device is determined from the CHRID of the device. The default is convert if the job CCSID does not match that of the underlying device; otherwise, standard is the default.

The possible values for this field are:

0	Same: Keep the current setting.
1	Standard: Do not perform conversion.
2	Convert: If the job CCSID does not match that of the display device and neither has the value 65535, perform the appropriate conversion on outgoing and incoming data. (See "Limitations and Restrictions" on page 292 for further details.)

Coexistence. Whether DSM coexists with other screen I/O methods, such as DDS- or UIM-coded interfaces, during the course of this application. Better performance can be achieved if coexistence is not required; the DSM APIs can assume the state of the device, for example, wide or normal mode.

The default is 1.

The possible values for this field are:

0	Same: Keep the current setting.
1	Coexist: Other screen I/O methods are used in conjunction with DSM.
2	Only: DSM is the only screen I/O method being used.

Color support. This field determines the color selection used when both a monochrome and a color display attribute are supplied. The default is select.

The possible values for this field are:

0	Same: Keep the current setting.
1	Mono: Always use the monochrome attributes specified regardless of the underlying device type.
2	Always use the color attributes specified regardless of the underlying device type.
3	Select: Select the appropriate attribute based on the underlying display type.

DBCS support. This field specifies whether the data being sent to the display contains DBCS data. This field affects, for example, how data is handled within sessions. For devices that do not support DBCS data, the default is standard; for DBCS-capable devices, it is mixed. If a value other than standard is specified for a device that does not support DBCS data, a CPFA306 error will occur.

The possible values for this field are:

0	Same: Keep the current setting.

1 Standard: Always handle data as single-byte characters.

2 Only: Data contains double-byte characters only. SO/SI control characters must enclose the data; these are not implicitly added.

3 Either: Data contains either double-byte or single-byte characters, but not both. SO/SI control characters must enclose the DBCS part of the data, unless a graphic DBCS value is passed. In this case, extended ideographic attributes must enclose the data, which can be written using the QsnWrtDta API (see "Write Data (QsnWrtDta) API" on page 155).

4 Mixed: Data may contain both single- and double-byte characters. All double-byte character strings within the data must be enclosed by SO/SI control characters. Graphic DBCS data must be enclosed by extended ideographic attributes as described for the preceding value.

Display file. The name of the display file to be used. The first 10 characters contain the file name, and the last 10 characters contain the library name. By specifying a display file, you can, for example, direct the input/output of different DSM environments to different devices, and you can specify the restore display parameter on the display file at creation time. The default for this field is blanks, which indicate that the system-supplied display file should be used.

If a file name is specified, the file must contain a record named USRRCD. This record must have the USRDFN keyword specified.

Exit routine to call when low-level environment changed. Exit routine to call when the low-level environment is changed through the QsnChgEnv or QsnSetEnvWinMod API. For a description of the parameters passed to this routine, see "Change Low-Level Environment Exit Routine" on page 14. Specify NULL for this field if no exit routine is required.

Exit routine to call when low-level environment deleted. Exit routine to call when the low-level environment is deleted through the QsnDltEnv API. The exit routine will be called before the environment itself is deleted. For a description of the parameters passed to this routine, see "Delete Low-Level Environment Exit Routine" on page 14. Specify NULL for this field if no exit routine is required.

Invite active. This field determines whether each write that is sent out causes the device to be invited. After the device is invited, the user is able to enter data, but the application can continue processing instead of waiting for input. When the application is ready for the input, it can call QsnReadInvited, which will issue a read from invited device operation.

The invite active flag is ignored by the read APIs, like QsnReadMDT. If the operation is indirect, then the read command will be added to the command buffer. This command buffer can be passed into the QsnReadInvited API, where it will be sent out before the read from invited device is done. If the operation is direct, then the read command will be sent in a data stream to the device. The new data stream will cause the invite to be retracted. A read with wait will be performed.

If the display file value is specified for the environment, the INVITE keyword must be specified on the USRRCD record of the specified display file.

The timeout value for the read from invited device operation can be controlled, by supplying a display file for the environment. The WAITRCD parameter on the CRTDSPF command can be set to the desired timeout value. See the return value for the QsnReadInvited API to determine the return value if the read from invited device operation times out.

The possible values for this field are:

0 Same: Keep the current setting.

1 Not active: The INVITE keyword, if specified in the display file, will be ignored. A normal read of the device will be performed.

2 Active: Any request to get data from the display will result in a read from invited device being issued.

For more information on invite processing, see the Application Display Programming book.

Prevent override. This field determines whether the display file used by DSM can be overridden or not. This value is ignored if the display file value is not specified.

The possible values for this field are:

0 Same: Keep the current setting.

1 Do not prevent override: When opened, the display file will allow overrides. For create and initialize operations, this is the default.

2 Prevent overrides: When the display file is opened, the flag to block overrides will be set.

User data associated with the environment. A pointer to any data the user wants to associate with this environment. This field can be used by the programmer to attach information to the low-level environment that can be of any format. For example, if multiple environments are being used, a list of the fields currently defined in an environment could be associated with the environment through this pointer. Specify NULL for this field if you do not have any user data.

Target device. The program device name of the target display device for the environment. This parameter must be specified with a value of *REQUESTER, which is the default.

Exit Routine Error Handling

If an exception occurs during the processing of an exit routine, the exception is ignored and processing continues. A CPFA318 will be issued as a diagnostic message only. You can explicitly handle errors in an exit routine by adding an exception handler to the routine.

Change Low-Level Environment Exit Routine

This exit routine, if specified on the user extension information, is called after the environment is changed through the QsnChgEnv or QsnSetEnvWinMod API. The following parameter is passed to the exit routine:

Change Low-Level Environment Exit Routine Parameter

Low-level environment handle
INPUT; BINARY(4)

A handle for the environment that was changed.

Delete Low-Level Environment Exit Routine

This exit routine, if specified on the user extension information, is called before the environment is deleted. The following parameter is passed to the exit routine:

Delete Low-Level Environment Exit Routine Parameter

Low-level environment handle
INPUT; BINARY(4)

A handle for the environment that was deleted.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA314 E	Memory allocation error.
CPFA31E E	Required parameter &1 omitted.
CPFA327 E	Low level environment description value incorrect.
CPFA344 E	The file &2 in library &3 is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Delete Low-Level Environment (QsnDltEnv) API

Required Parameter:

1 Low-level environment handle

Input Binary(4)

Omissible Parameter:

2 Error Code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Delete Low-Level Environment (QsnDltEnv) API deallocates an operating environment for low-level interface routines.

Authorities and Locks

None

Required Parameter

Low-level environment handle

INPUT; BINARY(4)

The low-level environment to be deallocated.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA317 E	Cannot deallocate memory dynamically.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Initialize Low-Level Environment Description (QsnInzEnvD) API

Required Parameter Group:	
1	Low-level environment description
Output	Char(*)
2	Length of low-level environment description
Input	Binary(4)
Omissible Parameter:	
3	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Initialize Low-Level Environment Description (QsnInzEnvD) API initializes a low-level environment description with default values. Unless otherwise specified in the low-level environment description in "Format of the Low-Level Environment Description" on page 10, pointer fields are set to the null pointer, numeric fields to 0, character flag fields to 0, and other character fields to blanks. For example, the default value for the color support field is 3, so this field will be set to 3.

Authorities and Locks

Display file authority
*USE

Display file library authority
*USE

Exit routine authority
*EXECUTE

Required Parameter Group

Low-level environment description
OUTPUT; CHAR(*)

The low-level environment description to be initialized.

Length of low-level environment description
INPUT; Binary(4)

The length of the low-level environment description parameter. This parameter must be specified as 16, 36 or 38.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA344 E	The file &2 in library &3 is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Query 5250 (QsnQry5250) API

Required Parameter Group:

1 Receiver variable

Output Char(*)

2 Length of receiver variable

Input Binary(4)

Omissible Parameter:

3 Error Code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Query 5250 (QsnQry5250) API is used to retrieve the results of the Query 5250 command for the current device. The Query 5250 command returns device and controller attributes for the current device, such as whether wide mode and graphical user interface (GUI) are supported.

Authorities and Locks

None

Restrictions

This command is not supported by all control units. A query status of 3 indicates if the query failed.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The receiver variable that is to receive the result of the query. You can specify that the size of the area be smaller than the format requested as long as you specify the length of the receiver variable parameter correctly. As a result, the API returns only the data the area can hold. The format of the data returned is shown in "Format of the Query Data."

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. If the length is larger than the size of the receiver variable, the results are unpredictable. The minimum length is 8 bytes.

The API returns as much information as it can fit in this length. If the available information is longer, it is truncated. If the available information is shorter, the unused output is unchanged; whatever is already stored in that space remains there. To determine how much information the API actually returns in response to this call, see the bytes returned field. To determine how much information the API could return if space were available, see the bytes available field.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Query Data

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(1)	Query status
9	9	BINARY(2)	Work station control unit

Offset		Type	Field
Dec	Hex		
11	B	CHAR(3)	Code Level
14	E	CHAR(16)	Reserved
30	1E	CHAR(1)	Work station type code
31	1F	CHAR(4)	Machine type code
35	23	CHAR(3)	Model number
38	26	CHAR(1)	Keyboard ID
39	27	CHAR(1)	Extended keyboard ID
40	28	CHAR(1)	PC keyboard ID
41	29	CHAR(4)	Serial number
45	2D	BINARY(2)	Maximum input fields
47	2F	CHAR(2)	Control unit customization
48	30	CHAR(1)	Reserved
50	32	CHAR(12)	Device capabilities
62	3E	CHAR(1)	Grid buffers
63	3F	CHAR(1)	Type of grid line support
64	40	CHAR(1)	Reserved
65	41	CHAR(1)	Images or faxes
66	42	CHAR(1)	Image or fax scaling granularity
67	43	CHAR(1)	Image or fax rotating granularity
68	44	CHAR(1)	Image or fax support
69	45	CHAR(1)	Invisible tags
70	46	CHAR(2)	Reserved

Field Descriptions

Further details on the fields listed can be found in the 5494 Remote Control Unit Functions Reference, SC30-3533, manual.

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Code Level. Identifies the code release level.

Control unit customization. Indicates customization parameters for the control unit as:

Byte 0

- Bit 0: Indicates that the iSeries server can send a 5250 WSC Customization command when set on
- Bit 1: Indicates that the iSeries server can send a 5250 Query Station State command when set on
- Bit 2: Indicates that the iSeries server can send a 5250 Workstation Customization command to select the SBA code returned in READ commands for displays with ideographic extended attributes when set on.
- Bit 3: Indicates that the 5250 Workstation Customization command may be either 6 bytes or greater than 8 bytes in length when set on.
- Bits 4-7: Reserved

Byte 1: Reserved

Device capabilities. Defines the operating capabilities of the designated device as:

Byte 0

- Bits 0-1: Indicate Row 1/Column 1 support as:

B'00'	No support
B'01'	limited support

- Bit 2: Indicates the Read MDT Alternate command is supported when set on
- Bit 3: Indicates the work station and control unit have PA1 and PA2 support when set on
- Bit 4: Indicates the work station and control unit have PA3 support when set on
- Bit 5: Indicates the work station and control unit have cursor select support when set on
- Bit 6: Indicates the work station and control unit have move cursor order support when set on
- Bit 7: Indicates the Read Modified Immediate Alternate command is supported when set on

Byte 1—display screen capabilities

- Bits 0-3: Define screen size as:

B'0001'	24 x 80
B'0011'	24 x 80 or 27 x 132

- Bit 4: Indicates selector light pen (SLP) is supported when set on
- Bit 5: Indicates magnetic stripe reader (MSR) is supported when set on
- Bits 6-7: Define color support as:

B'00'	Monochrome display
B'01'	Color support

Byte 2

- Bit 0: Indicates Text Symbols support when set on
- Bit 1: Indicates work station and control unit have extended primary attribute
- Bits 2-4: Indicate Office Editor/Text support as:

B'000'	No Office Editor/Text support
B'001'	single language Office Editor/Text support
B'010'	dual language Office Editor/Text support

- Bit 5: Indicates work station and control unit have extended primary attribute support in data processing (DP) mode (WEA order) when set on
- Bits 6-7: Indicates extended foreground color attribute support

B'01'	Available in DP mode. Fourteen colors are defined, but only seven are available. The other seven colors are mapped into the available colors.
B'10'	Available in DP mode. Fourteen colors are supported.

Byte 3

- Bits 0-2: Indicate ideographic capability as:

B'000'	No ideographic capability
B'001'	Ideographic capability for presentation screen only
B'010'	Ideographic data type and presentation screen ideographic capability

- Bits 3-5: Indicate bidirectional support as:

B'000'	No bidirectional capability
B'001'	Bidirectional capability

- Bit 6: Ideographic
- Bit 7: Indicates CCSID-based I/O is supported when set on.

Byte 4

- Bits 0-2: Indicate graphics capability as:

B'000'	No graphics capability
B'001'	5292-style graphics
B'010'	GDDM(R)-OS/2(R) Link Graphics

- Bit 3: Indicates extended 3270 data stream capability when set on
- Bit 4: Indicates a pointer device is available when set on
- Bit 5: Indicates that GUI-like characters are available when set on
- Bit 6: Indicates the control unit supports enhanced user interface commands and field control words (FCWs) when set on.

  The commands include:

  Create Window

  Unrestricted Cursor Movement

  Remove GUI Window

  Remove All GUI Constructs

  Read Screen To Print

  Read Screen To Print With Extended Attributes

  Write Error Code To Window

  Save Partial Screen

  Restore Partial Screen

  Define Selection Field

  Remove GUI Selection Field

Define Scroll Bar

Remove GUI Scroll Bar

The FCWs include:

Continued

Cursor Progression

Highlighted

Pointer Device Selection

- Bit 7: Indicates Write Error Code To Window command is supported when set on

Byte 5

- Bit 0: Indicates the Write Data and Programmable Mouse Buttons structured field commands, the Word Wrap FCW, and Ideographic Continued entry fields are supported when set on
- Bit 1: Indicates this is a GUI device which will use all-points-addressable constructs for windows, selection fields, and scroll bars, when set on
- Bits 2-7: Reserved

Byte 6: Reserved

Bytes 7-8:

- Bit 0-13: Reserved
- Bit 14-15: 5250 fax or image support

B'00'	No 5250 image or fax support
B'01'	Support for seven formats: • TIFF – No compression – CCITT Group 3 fax one-dimensional, modified-Huffman run-length encoding – CCITT Group 3 fax compression – CCITT Group 4 fax compression – PackBits run-length encoding • PCX monochrome format • Stand-alone CCITT Group 3 fax compression
B'11'	Support for the seven previous formats, plus five additional formats: • IOCA – IBM MMR algorithm – No compression – CCITT Group 3 fax one-dimensional, modified-Huffman run-length encoding – CCITT Group 3 fax compression – CCITT Group 4 fax compression

Byte 9: Reserved for use by PC emulators to indicate additional 5250 image or fax formats supported

Byte 10:

- Bit 0: Indicates printer type as:

B'0'	SCS printer
B'1'	IPDS printer

- Bits 1-7: Reserved

Byte 11: Reserved

Extended keyboard ID. The device code for extended 5250 keyboards.

Grid buffers. The number of grid buffers that are available in the device.

X'00'	Not grid-capable.

Images or faxes. The number of images or faxes that can be presented on a display screen.

X'00'	No 5250 image or fax support
X'01-FE'	Number allowed
X'FF'	Variable, dependent on the size of the image or fax

Image or fax scaling granularity.

X'00'	No 5250 image or fax support
X'01'	Support for scaling percentages from 3% to 400%. No scroll-bar scaling, fill scaling, no change scaling, increment and decrement
X'02-7E'	Reserved
X'7F'	Support for continuous scaling
X'80-FF'	Reserved for use by 5250 PWS emulators

Image or fax rotating granularity.

X'00'	No 5250 image or fax support
X'01'	Support for rotating of 0, 90, 180, and 270 degrees
X'02-7E'	Reserved
X'7F'	Support for continuous rotation
X'80-FF'	Reserved for use by 5250 PWS emulators

Image or fax support.

X'00'	No 5250 image or fax support	
	B'0'	Pop-up and pull-down windows that were written after image or fax are presented over image or fax data when set on
	B'1'	This device supports transparent mode.
	B'2-7'	Reserved

Invisible tags. Defines more device capabilities of the designated device as:
- Bits 0-5: Reserved
- Bit 6: EBCDIC-to-ASCII translation. This is used by workstation gateway devices.
- Bit 7: True transparency.

Keyboard ID. Reserved. This field is set to X'00'.

Machine type code. An EBCDIC code for the machine type.

Maximum input fields. The maximum number of input fields available (256).

Model number. An EBCDIC code for the machine model number.

PC keyboard ID. Device code for PC keyboards attached to a 5250 work station (X'00' for nonprogrammable work stations).

Query status. The status of the 5250 query data. The possible values are:

DSM_5250Q_YES (1)	Query information successfully retrieved.
DSM_5250Q_NO (2)	Query cannot be issued for the device. This occurs when the device configuration specifies that the query command should not be issued against the device.
DSM_5250Q_FAIL (3)	Query command failed. Default values are supplied based on the device type. This occurs, for example, when the controller does not support the query command.

Reserved. An ignored field.

Serial number. Field for device serial number. This field is set to zero for a work station with no serial number.

Type of grid line support.

X'00'	No grid line support
X'01'	Type 1 grid line support including support for grid line commands

Work station control unit. The type of control unit.

Work station type code. The workstation type. The value is X'01' for display station.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Query Color Support (QsnQryColorSup) API

Omissible Parameter Group:	
1	Color indication
Output	Char(1)
2	Low-level environment handle
Input	Binary(4)
3	Error code
I/O	Char(*)
Returned Value:	
	Color indication
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Query Color Support (QsnQryColorSup) API determines whether the current display device supports color or not.

Authorities and Locks

None

Omissible Parameter Group

Color indication

OUTPUT; CHAR(1)

Whether the device supports color or not. This information will be set based on the results of the 5250 Query command, if the display supports it; otherwise, certain defaults are assumed. See "Device Support" on page 291 for details. The possible values are:

0	Device does not support color
1	Device supports color

Low-level environment handle

INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Color indication
OUTPUT; BINARY(4)

This API returns the value for the color indication parameter if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Query Display Mode Support (QsnQryModSup) API

Required Parameter:

1 Display mode

Input Char(1)

Omissible Parameter Group:

2 Mode indication

Output Char(1)

3 Low-level environment handle

Input Binary(4)

4 Error code

I/O Char(*)

Returned Value:

Mode indication

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Query Display Mode Support (QsnQryModSup) API determines if the current display device supports the given mode. Certain devices, like the 3486 and 3487, support 27x132 mode but can be switched by keystroke to turn off the wide capability. This will be reflected in the result returned by the QsnQryModSup API. Use this API to determine if a subsequent mode change request through the Clear Screen (QsnClrScr) API is valid. You can use the result of the Query 5250 (QsnQry5250) API to determine if the display is capable of supporting wide mode or not.

Authorities and Locks

None

Required Parameter

Display mode

INPUT; CHAR(1)

The display mode for which to query support. The possible values are:

3	24x80 mode
4	27x132 mode

Omissible Parameter Group

Mode indication

OUTPUT; CHAR(1)

Whether the device allows the specified mode or not. The possible values are:

0	Device does not support the mode
1	Device supports the mode

Low-level environment handle

INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Mode indication

OUTPUT; BINARY(4)

This API returns the value for the mode indication parameter if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA322 E	Incorrect display mode &1 specified.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Restore Screen (QsnRstScr) API

Required Parameter:	
1	Input buffer containing saved data
Input	Binary(4)
Omissible Parameter Group:	
2	Command buffer handle
Input	Binary(4)
3	Low-level environment handle
Input	Binary(4)
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Restore Screen (QsnRstScr) API restores the state of the display as saved with an indirect command. The display will be restored using the data contained in the input buffer given by parameter 1. If an indirect operation is specified, the resulting command buffer will contain the Restore Screen command and the data to restore the screen. Additional commands can be added to the command buffer subject to the conditions described in "Restrictions."

This command corresponds directly to the 5250 Restore Screen or Restore Partial Screen command. See the 5250 data stream documentation for details.

Authorities and Locks

None

Restrictions

This command must be the last command in the command buffer except when GUI support is used. In this case, other input commands may follow.

Required Parameter

Input buffer containing saved data
INPUT; BINARY(4)

An input buffer that contains the result of an indirect QsnSavScr operation. The data will be copied from this input buffer and used for the restore screen operation.

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

If this parameter is omitted or specified as 0, this is a direct operation and the screen is restored immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA316 E	Saved data not valid.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Display Mode (QsnRtvMod) API

Omissible Parameter Group:	
1	Display mode
Output	Char(1)
2	Low-level environment handle
Input	Binary(4)
3	Error code
I/O	Char(*)
Returned Value:	
	Display mode
Output	Char(1)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Display Mode (QsnRtvMod) API returns the current display mode.

Authorities and Locks

None

Omissible Parameter Group

Display mode
OUTPUT; CHAR(1)

The current display mode. The possible values are:

3	Device is in 24x80 mode
4	Device is in 27x132 mode

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Display mode
OUTPUT; CHAR(1)

This API returns the value for the display mode parameter, or 0 if an error occurs during processing.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA322 E	Incorrect display mode &1 specified.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Low-Level Environment Description (QsnRtvEnvD) API

Required Parameter Group:

1 Low-level environment description

Output Char(*)

2 Length of low-level environment description

Input Binary(4)

Omissible Parameter Group:

3 Low-level environment handle

Input Binary(4)

4 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Low-Level Environment Description (QsnRtvEnvD) API returns the description corresponding to the specified low-level environment.

Authorities and Locks

None

Required Parameter Group

Low-level environment description
Output; CHAR(*)

The variable that contains the low-level environment description when the QsnRtvEnvD API has completed. The format of the data returned is shown in "Format of the Data Returned."

Length of low-level environment description
INPUT; Binary(4)

The length of the low-level environment description parameter. The minimum length is 8. If the length is larger than the size of the receiver variable, the results are not predictable. The API returns as much information as it can fit in this length. It the available information is longer, it is truncated. If the available information is shorter, the unused output is unchanged; whatever is already stored in that space remains there. To determine how much information the API actually returns in response to this call, see the bytes returned field. To determine how much information the API could return if space were available, see the bytes available field.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment for which the description should be retrieved. If this parameter is omitted or specified as 0, the description for the default low-level environment is retrieved.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Data Returned

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(*)	Environment description The format of the remaining data returned is shown in "Format of the Low-Level Environment Description" on page 10.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C24 E	Length of the receiver variable is not valid.

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Low-Level Environment User Data (QsnRtvEnvDta) API

Required Parameter:

1 Low-level environment handle

Input Binary(4)

Omissible Parameter Group:

2 User data pointer

Output PTR(SPP)

3 Error code

I/O Char(*)

Returned Value:

User data pointer

Output PTR(SPP)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Low-Level Environment User Data (QsnRtvEnvDta) API returns a pointer to the user data for the given low-level environment.

Authorities and Locks

None

Required Parameter

Low-level environment handle

INPUT; BINARY(4)

A handle for the low-level environment for which the user data should be returned. If this parameter is omitted or specified as 0, the default low-level environment is used.

Omissible Parameter Group

User data pointer

OUTPUT; PTR(SPP)

A pointer to the user data, as specified on the low-level environment description, for the given low-level environment.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

User data pointer

OUTPUT; PTR(SPP)

This API returns the value for the user data pointer parameter, or the null pointer if an error occurs.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1F E	Pointer is not on a 16 byte boundary.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Low-Level Environment Window Mode (QsnRtvEnvWinMod) API

Required Parameter Group:	
1	Window mode description
Output	Char(*)
2	Length of window mode description
Input	Binary(4)
Omissible Parameter Group:	
3	Low-level environment handle
Input	Binary(4)
4	Error Code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Low-Level Environment Window Mode (QsnRtvEnvWinMod) API queries the state of the window mode for the low-level interfaces.

Authorities and Locks

None

Required Parameter Group

Window mode description
OUTPUT; CHAR(*)

The field in which the window mode description should be stored. The format of the data returned in this field is described in "Format of the Data Returned" on page 37.

Length of window mode description
INPUT; BINARY(4)

The length of the window mode description parameter. If the length is larger than the size of the receiver variable, the results are not predictable. The minimum length is 8. The API returns as much information as it can fit in this length. If the available information is longer, it is truncated. If the available information is shorter, the unused output is unchanged; whatever is already stored in that space remains there. To determine how much information the API actually returns in response to this call, see the bytes returned field. To determine how much information the API could return if space were available, see the bytes available field.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Data Returned

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(1)	Window mode
9	9	CHAR(*)	Window mode description

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Window mode. Whether window mode is enabled or disabled. The possible values are:

0	Window mode is disabled.
1	Window mode is enabled.

Window mode description. The format of the remaining data returned is shown in "Format of the Window Mode Description" on page 49.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.

Message ID	Error Message Text
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Screen Dimensions (QsnRtvScrDim) API

Omissible Parameter Group:

1	Number of rows
Output	Binary(4)
2	Number of columns
Output	Binary(4)
3	Low-level environment handle
Input	Binary(4)
4	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Screen Dimensions (QsnRtvScrDim) API retrieves the current dimensions of the screen. You must specify either the number-of-rows or the number-of-columns parameter, or a CPFA31E message will be issued.

Authorities and Locks

None

Omissible Parameter Group

Number of rows
OUTPUT; BINARY(4)

The current height of the screen. This information will be set based on the current display size.

Number of columns
OUTPUT; BINARY(4)

The current width of the screen. This information will be set based on the current display size.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Roll Down (QsnRollDown) API

Required Parameter Group:	
1	Number of lines to roll
Input	Binary(4)
2	Top row of roll area
Input	Binary(4)
3	Bottom row of roll area
Input	Binary(4)
Omissible Parameter Group:	
4	Command buffer handle
Input	Binary(4)
5	Low-level environment handle
Input	Binary(4)
6	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Roll Down (QsnRollDown) API rolls the screen down a given number of lines within the roll area specified. The following conditions cause a CPFA315 error to occur:

- A top row of zero
- A bottom row greater than the number of display lines
- A top row greater than or equal to the bottom row
- A roll area greater than the bottom row minus the top row

This API corresponds directly to the 5250 Roll command. See the 5250 data stream documentation for details.

Restrictions

The following considerations apply to the QsnRollDown API:

- Lines vacated due to a roll are not cleared to nulls.
- The command does not change the format table, and so, should be avoided when it could produce discrepancies between the format table and the display.
- Data rolled out of the roll area are lost.

Authorities and Locks

None

Required Parameter Group

Number of lines to roll
INPUT; BINARY(4)

The number of lines to roll the designated area down by.

Top row of roll area
INPUT; BINARY(4)

The line number defining the top line of the area that will participate in the roll.

Bottom row of roll area
INPUT; BINARY(4)

The line number defining the bottom line of the area that will participate in the roll.

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

If this parameter is omitted or specified as 0, this is a direct operation and the screen is rolled down immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA315 E	Roll parameters not valid.
CPFA31E E	Required parameter &1 omitted.

Message ID	Error Message Text
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Roll Up (QsnRollUp) API

Required Parameter Group:

1 Number of lines to roll

Input Binary(4)

2 Top row of roll area

Input Binary(4)

3 Bottom row of roll area

Input Binary(4)

Omissible Parameter Group:

4 Command buffer handle

Input Binary(4)

5 Low-level environment handle

Input Binary(4)

6 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

Usage Notes

The Roll Up (QsnRollUp) API is identical in function to the QsnRollDown API, except the lines are rolled up instead of down.

Authorities and Locks

None

Required Parameter Group

Number of lines to roll
INPUT; BINARY(4)

The number of lines to roll the designated area up by.

Top row of roll area
INPUT; BINARY(4)

The line number defining the top line of the area that will participate in the roll.

Bottom row of roll area
INPUT; BINARY(4)

The line number defining the bottom line of the area that will participate in the roll.

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

If this parameter is omitted or specified as 0, this is a direct operation and the screen is rolled up immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA315 E	Roll parameters not valid.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.

Message ID	Error Message Text
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Save Screen (QsnSavScr) API

Omissible Parameter Group:

1	Saved data command buffer handle
Output	Binary(4)
2	Command buffer handle
Input	Binary(4)
3	Low-level environment handle
Input	Binary(4)
4	Error code
I/O	Char(*)

Returned Value:

Saved data command buffer handle

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Save Screen (QsnSavScr) API saves the current state of the display. If this is a direct operation, the API creates a command buffer that contains the operations used to restore the screen state and returns a handle for this buffer. When a direct save screen operation is issued, the saved screen data is returned in a command buffer. This command buffer contains the Restore Screen command along with the data to restore the screen. Additional commands can be added to the command buffer as described in "Restrictions" on page 29 under the Restore Screen (QsnRstScr) API. The screen can be restored by sending this command buffer using the Put Command Buffer (QsnPutBuf) API.

When an indirect save screen operation is issued, the Save Screen command is stored in the command buffer and is considered an input operation. The command can be issued to the screen only through the Put Command Buffer and Perform Get (QsnPutGetBuf) API. The saved data will be returned in the input buffer parameter specified for the QsnPutGetBuf API. The screen can subsequently be restored by specifying this input buffer on the QsnRstScr API.

This command corresponds directly to the 5250 Save Screen (when the underlying control unit supports it) or Save Partial Screen command. See the 5250 data stream documentation for details.

Authorities and Locks

None

Restrictions

This command must be the last command in the command buffer, except when GUI support is available. In this case, other non-input commands may follow.

Omissible Parameter Group

Saved data command buffer handle
OUTPUT; BINARY(4)

The variable that will contain the command buffer handle for the restore screen operation if this is a direct operation.

For an indirect operation, the result of the save screen will be returned in the input buffer of a subsequent input operation, which can be used to restore the screen using the QsnRstScr operation.

Command buffer handle
INPUT; BINARY(4)

If this parameter is omitted or specified as 0, this is a direct operation and the screen is saved immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Saved data
OUTPUT; BINARY(4)

For a successful operation, this API returns the value for the saved data parameter if this is a direct operation; otherwise, it returns zero. For an unsuccessful operation, it returns -1.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA314 E	Memory allocation error.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.

API introduced: V2R3

Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API

Required Parameter:

1 Enable window mode

Input Char(1)

Omissible Parameter Group:

2 Previous window mode setting

Output Char(1)

3 Window mode description

Input Char(*)

4 Length of window mode description

Input Binary(4)

5 Low-level environment handle

Input Binary(4)

6 Error Code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API enables or disables the window mode for the low-level environment. Use this API to affect how row and cursor positions specified on low-level interface operations are interpreted for operations to the given low-level environment. Additional details regarding windows can be found in Window Services APIs.

When window mode is enabled, screen locations and cursor positions specified and retrieved in the low-level interface routines are interpreted relative to the logical window area defined. The logical window area is treated as a logical screen in terms of validity of cursor and data starting positions. If an attempt is made to write data or define a field that starts outside of the current window area, a CPFA31D error is issued for that particular API. If data or a field is written to the window that exceeds the window boundary, no error condition occurs and the data or field is truncated.

The size of the logical window area can be determined through the Retrieve Low-Level Environment Window Mode (QsnRtvEnvWinMod) API. When window mode is enabled, screen addresses must be explicitly specified for APIs such as the QsnWrtDta API in order for the address to be interpreted relative to the window area. If a screen address is not specified, the current display address will be used as an

absolute screen address irrespective of the current window mode. This is because the current screen address is not tracked by DSM, but is handled by the control unit.

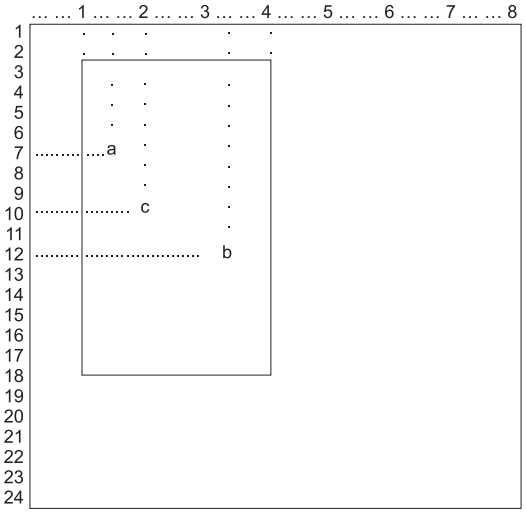
When window mode is enabled, APIs where a cursor position is specified, such as the QsnSetFld API, issue a CPFA307 error if the given cursor position is outside the bounds of the current window area. For APIs that return a cursor position, such as the Get Cursor Address (QsnGetCsrAdr) API, both the row and column returned will be -1 if the cursor screen location is outside of the current window area, 0 if the cursor is on the top or left border, or the number of screen rows or columns plus 1 if the cursor is on the bottom or center border of the window area, respectively. The following low-level APIs are affected by the window mode:

QsnRtvReadAdr
QsnRtvFldInf
QsnGetCsrAdr
QsnGetCsrAdrAID
QsnSetOutAdr
QsnWrtDta
QsnWrtSFMaj
QsnWrtTDta
QsnWrtExtAtr
QsnWrtPad
QsnWrtPadAdr
QsnSetFld
QsnSetCsrAdr
QsnInsCsr
QsnSetErr

The actual screen location used for a screen I/O operation is calculated using the formula base+offset=actual, where base is the upper left row/column location of the window border (0 for full screen) if offset is positive and the lower center row/column location of the window border (screen height/width plus 1 for full screen), if offset is negative, offset is the row/column specified on the API, and actual is the actual screen location. For example, if the window area were defined to be from row 3, column 10 with 15 rows and 30 columns, as shown in Window Area (page 48), then an attempt to position the cursor through the QsnSetCsrAdr API at row 4, column 5 would actually position the cursor on the screen at row 7, column 15, as indicated by the letter a in Window Area (page 48). Specifying row 9, column -7 would position the cursor on the screen at row 12, column 34 (b in Window Area (page 48)). An attempt to position the cursor at error issue a CPFA307 error row 16, column 5 would result error (Screen position &1,&2 outside of display or window area). since this position is outside the bounds of the current window area. Given the same window area description, a call to the QsnRtvFldInf API specifying an input buffer containing a field read from row 10, column 20 on the actual screen (the c in Window Area (page 48)), would return a field row and column location of 7 and 10 respectively. A call to the QsnGetCsrAdr API would return -1,-1 if the cursor were located outside of the window area, such as in row 18 column 32.

Enabling or disabling the window mode does not affect any data currently displayed on the screen or the behavior of any commands stored previously in a command buffer. For example, if the window mode was enabled as described above and the QsnSetCsrAdr API was called as an indirect operation specifying row 4 and column 5, the cursor position command stored in the command buffer would reflect the current window mode. Whenever that command buffer is written out, the cursor would always be set on the screen at row 6, column 14, regardless of whether or not window mode was disabled or changed at the point when the command buffer was written to the screen.

Window Area



RBAFX646-0

Authorities and Locks

None

Required Parameter

Window mode

INPUT; CHAR(1)

Whether window mode should be enabled or disabled. The possible values are:

0	Disable window mode
1	Enable window mode

Window mode is initially disabled.

Omissible Parameter Group

Previous window mode setting

OUTPUT; CHAR(1)

Whether window mode was enabled or disabled prior to this API being called. The possible values returned are:

0	Window mode was disabled prior to this API being called
1	Window mode was enabled prior to this API being called

Window mode description

INPUT; CHAR(*)

The window mode description. If this parameter is omitted or the length parameter is specified as 0, and window mode is to be enabled, the values from the previous window mode setting will be used. If no such values exist, the current screen size will be used. The window area described must fall within the bounds of the current screen size. in a CPFA307 error (Screen position &1,&2 outside of display or window area). The format of this field is shown in “Format of the Window Mode Description” on page 49.

This parameter is ignored if window mode is to be disabled.

Length of window mode description

INPUT; BINARY(4)

The length of the window mode description parameter. If this parameter is specified, it must be either 0, in which case the window mode description parameter is ignored, or exactly 13 bytes in which case the window mode description is required.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Window Mode Description

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Attribute column indication
1	1	BINARY(4)	Upper left row of window area border
5	5	BINARY(4)	Upper left column of window area border
9	9	BINARY(4)	Number of rows in window area
13	D	BINARY(4)	Number of columns in window area

Field Descriptions

Attribute column indication. Whether the column containing the left border of the logical window is an attribute column. Operations such as QsnWrtDta can specify column 1 for the data location and specify a leading attribute. In this case the data will be written to the first column of the window area and the attribute will be written to the column containing the logical window border. If the attribute column is not specified for the window area, such an operation would result in a CPFA31D error (Attempt to write outside of window area.). This column would also be used to insert the leading attribute when line wrapping occurs within the window.

The allowable values are:

0	No attribute column
1	Attributes can be written to column containing left logical window border

Number of columns in window area. The number of columns in the window area.

Number of rows in window area. The number of rows in the window area.

Upper left column of window area border. The column location of the leftmost column in the window area. This parameter must be a value between 0 and the screen width inclusive.

Upper left row of window border. The row location of the upper border of the logical window area. This parameter must be a value between 0 and the screen height inclusive.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA324 E	Window area definition is incorrect.
CPFA32A E	Window mode indicator must be '0' or '1'.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Buffer Manipulation and Query APIs

The buffer manipulation and query APIs are used to create, query, and manipulate input and command buffers that interact with the screen.

The two buffer types used by the low-level interfaces are command and input. A command buffer can be used to accumulate a sequence of low-level commands without performing an I/O operation. The entire buffer can be written out at once in a single I/O operation. See "Direct and Indirect Operations" on page 291 for further discussion of command buffers. When an input operation is performed, you can specify an input buffer as the target of the operation. The input results are placed in this buffer, which can then be queried through several interfaces. You may query for individual pieces of information, such as the AID byte from the input operation using the Retrieve AID Code on Read (QsnRtvReadAID) API or for multiple pieces of information using the Retrieve Read Information (QsnRtvReadInf) API.

Command and input buffers can be used across activation groups if the activation group in which the buffer was created still exists.

The buffer manipulation and query interfaces are:

- "Clear Buffer (QsnClrBuf) API" on page 51 (QsnClrBuf) clears all commands or data in a buffer resets its state.
- "Copy Buffer (QsnCpyBuf) API" on page 53 (QsnCpyBuf) copies the contents of one buffer to another.
- "Create Command Buffer (QsnCrtCmdBuf) API" on page 55 (QsnCrtCmdBuf) creates a command buffer to accumulate low-level commands.
- "Create Input Buffer (QsnCrtInpBuf) API" on page 57 (QsnCrtInpBuf) creates an input buffer to receive input results.
- "Delete Buffer (QsnDltBuf) API" on page 59 (QsnDltBuf) deletes a buffer.
- "Put Command Buffer (QsnPutBuf) API" on page 60 (QsnPutBuf) sends the commands in a command buffer to the screen.
- "Put Command Buffer and Perform Get (QsnPutGetBuf) API" on page 62 (QsnPutGetBuf) sends the commands in a command buffer to the screen and performs a read operation.
- "Retrieve AID Code on Read (QsnRtvReadAID) API" on page 64 (QsnRtvReadAID) determines the Aid code for a given input operation.

- "Retrieve Available Data (QsnRtvAvailData) API" on page 65 (QsnRtvAvailData) copies the user"s data into an input buffer.
- "Retrieve Buffer Data Length (QsnRtvBufLen) API" on page 67 (QsnRtvBufLen) returns the length of data in a buffer.
- "Retrieve Buffer Size (QsnRtvBufSiz) API" on page 68 (QsnRtvBufSiz) returns the size of a buffer.
- "Retrieve Cursor Address on Read (QsnRtvReadAdr) API" on page 70 (QsnRtvReadAdr) retrieves the cursor position at the completion of an input operation.
- "Retrieve Field Information (QsnRtvFldInf) API" on page 72 (QsnRtvFldInf) returns information about a particular field in an input buffer.
- "Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API" on page 75 (QsnRtvDtaLen) retrieves the number of data bytes in an input buffer after an input operation.
- "Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API" on page 76 (QsnRtvFldDtaLen) retrieves the number of bytes of field data after an input operation.
- "Retrieve Number of Bytes Read from Screen (QsnRtvReadLen) API" on page 78 (QsnRtvReadLen) retrieves the number of data bytes read from the screen after an input operation.
- "Retrieve Number of Fields Read (QsnRtvFldCnt) API" on page 79 (QsnRtvFldCnt) retrieves the number of fields in an input buffer.
- "Retrieve Pointer to Data in Input Buffer (QsnRtvDta) API" on page 81 (QsnRtvDta) returns a pointer to the first byte of input data in an input buffer.
- "Retrieve Pointer to Field Data (QsnRtvFldDta) API" on page 82 (QsnRtvFldDta) returns a pointer to the first byte of field data in an input buffer.
- "Retrieve Read Information (QsnRtvReadInf) API" on page 84 (QsnRtvReadInf) returns information about the input operation.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Clear Buffer (QsnClrBuf) API

Required Parameter Group:

1 Buffer handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Clear Buffer (QsnClrBuf) API clears all commands or data and resets the state of the given buffer. This is the only API that clears or removes data in a buffer.

Authorities and Locks

None

Required Parameter

Buffer handle

INPUT; BINARY(4)

A handle for the buffer to be cleared. The storage associated with the buffer is not deallocated.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Copy Buffer (QsnCpyBuf) API

Required Parameter Group:	
1	Source buffer handle
Input	Binary(4)
2	Target buffer handle
Input	Binary(4)
Omissible Parameter:	
3	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Copy Buffer (QsnCpyBuf) API copies the contents of one buffer to another buffer. Both buffers must be the same type—command or input. If the target and source buffers are the same, no operation takes place and no error is reported.

If a target command buffer contains data, the data in the source buffer is appended to the target buffer. A CPFA301 error is issued if the target command buffer is not large enough to hold the contents of the source buffer and cannot be resized.

If input buffers are being copied, the target buffer must be empty. If the target input buffer is not large enough to hold the data from the source buffer, the data is truncated and no error is reported.

Authorities and Locks

None

Required Parameter Group

Source buffer handle
INPUT; BINARY(4)

A handle for the buffer from which data is to be copied. The contents of this buffer are not affected by this operation.

Target buffer handle
INPUT; BINARY(4)

A handle for the buffer to which data is to be copied.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA305 E	Cannot add operation to command buffer.
CPFA301 E	Command buffer is full.
CPFA313 E	Command buffer already contains an input operation.
CPFA31E E	Required parameter &1 omitted.
CPFA330 E	Buffer type mismatch.
CPFA331 E	Buffer handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Create Command Buffer (QsnCrtCmdBuf) API

Required Parameter:	
1	Initial command buffer size
Input	Binary(4)
Omissible Parameter Group:	
2	Increment amount
Input	Binary(4)
3	Maximum size
Input	Binary(4)
4	Command buffer handle
Output	Binary(4)
5	Error code
I/O	Char(*)
Returned Value:	
	Command buffer handle
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Create Command Buffer (QsnCrtCmdBuf) API creates a command buffer for use with low-level operations that accept a command buffer parameter.

Authorities and Locks

None

Required Parameter

Initial command buffer size
INPUT; BINARY(4)

The initial size of the command buffer, in bytes, to create. This parameter must be greater than 0 and less than the size of the underlying display file I/O buffer: approximately 4500 bytes for 24x80, 6300 bytes for 27x132, 8000 bytes for DBCS-capable displays, 8800 bytes for DBCS presentation screen-capable displays, and 16000 bytes for DBCS ideographic-capable displays.

Omissible Parameter Group

Increment amount
INPUT; BINARY(4)

The amount to increment the command buffer size by if there is not enough space to store a specified command. If this parameter is omitted or specified with a zero value, the buffer size will not be incremented and a CPFA301 error will be issued when there is no space in the buffer

to store a requested command. If an attempt is made to increment a command buffer to a size that exceeds the available memory resources or the size of the underlying display file I/O buffer, the increment will not take place and a CPFA301 error will be issued for that operation.

Maximum size
INPUT; BINARY(4)

The maximum size to increment the command buffer to when there is not enough space to store a specified command. If this parameter is nonzero, it must be greater than the initial command buffer size parameter, and less than the size of the underlying display file I/O buffer. If this parameter is omitted or specified with a zero value, no maximum value is assigned for the command buffer. If the buffer is to be incremented, it will be incremented until either there is no additional storage available or the command buffer exceeds the size of the display file I/O buffer. If the increment amount parameter is omitted or specified with a zero value, this parameter is ignored and the maximum size is the same as the initial command buffer size.

Command buffer handle
OUTPUT; BINARY(4)

The variable containing the handle for the command buffer created after the QsnCrtCmdBuf API has completed. The buffer state will be the same as that following a QsnClrBuf operation.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Command buffer handle
OUTPUT; BINARY(4)

This API returns the value for the command buffer handle parameter if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA312 E	Buffer size parameter error.
CPFA314 E	Memory allocation error.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Create Input Buffer (QsnCrtInpBuf) API

Required Parameter:

1 Input buffer size

Input Binary(4)

Omissible Parameter Group:

2 Increment amount

Input Binary(4)

3 Maximum size

Input Binary(4)

4 Input buffer handle

Output Binary(4)

5 Error code

I/O Char(*)

Returned Value:

Input buffer handle

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Create Input Buffer (QsnCrtInpBuf) API creates an input buffer for use with low-level commands that accept an input buffer parameter.

Authorities and Locks

None

Required Parameter

Input buffer size

INPUT; BINARY(4)

The size of the input buffer, in bytes, to create. This parameter must be greater than 0 and less than the size of the underlying display file I/O buffer: approximately 4500 bytes for 24x80, 6300 bytes for 27x132, 8000 bytes for DBCS-capable displays, 8800 bytes for DBCS presentation screen-capable displays, and 16000 bytes for DBCS ideographic-capable displays.

Omissible Parameter Group

Increment amount

INPUT; BINARY(4)

The amount to increment the buffer size by if there is not enough space to store a read operation. If this parameter is omitted or specified with a zero value, the buffer size is not be incremented and input data is truncated if there is not enough space.

Maximum size
INPUT; BINARY(4)

The maximum size to increment the input buffer to when there is not enough space to store the result of a read operation. If this parameter is nonzero, it must be greater than the initial input buffer size parameter, and less than the size of the underlying display file I/O buffer. If this parameter is omitted or specified with a zero value, no maximum value is assigned for the input buffer, if the buffer is to be incremented, it will be incremented until either there is no additional storage available or the input buffer exceeds the size of the display file I/O buffer. If the increment amount parameter is omitted or specified with a zero value, this parameter is ignored and the maximum size is the same as the initial input buffer size.

Input buffer handle
OUTPUT; BINARY(4)

The variable containing the handle for the created input buffer after the QsnCrtInpBuf API has completed. The buffer state becomes the same as that following a QsnClrBuf operation.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Input buffer handle
OUTPUT; BINARY(4)

This API returns the value for the input buffer handle parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA312 E	Buffer size parameter error.
CPFA314 E	Memory allocation error.

API Introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Delete Buffer (QsnDltBuf) API

Required Parameter Group:		
1	Buffer handle	
Input	Binary(4)	
Omissible Parameter:		
2	Error code	
I/O	Char(*)	
Returned Value:		
	Return code	
Output	Binary(4)	
Default Public Authority: *USE		
Service Program: QSNAPI		
Threadsafe: No		

The Delete Buffer (QsnDltBuf) API deletes a command or input buffer created with the Create Command Buffer (QsnCrtCmdBuf) or the Create Input Buffer (QsnCrtInpBuf) API, respectively. All storage associated with the buffer is deallocated.

Authorities and Locks

None

Required Parameter

Buffer handle

Input; BINARY(4)

A handle for the buffer to be deleted.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Put Command Buffer (QsnPutBuf) API

Required Parameter Group:

1 Command buffer handle

Input Binary(4)

Omissible Parameter Group:

2 Low-level environment handle

Input Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Put Command Buffer (QsnPutBuf) API sends the commands accumulated in a command buffer to the screen. This corresponds to a write operation to the display file. If the command buffer contains no data, the operation returns successfully, but no I/O operation is performed.

Authorities and Locks

None

Required Parameter

Command buffer handle

INPUT; BINARY(4)

A handle for the command buffer. The command buffer is not modified in any way as a result of this command.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA308 E	Attempt to write data past end of display.
CPFA309 E	Invalid cursor position in command buffer.
CPFA30A E	Field length &1 not valid.
CPFA30B E	Invalid starting address for field.
CPFA30C E	Maximum allowable number of fields exceeded.
CPFA316 E	Saved data not valid.
CPFA31B E	From position &1, &2 greater than to position &3, &4.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA338 E	Command buffer contains an input operation.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Put Command Buffer and Perform Get (QsnPutGetBuf) API

Required Parameter Group:	
1	Command buffer handle
Input	Binary(4)
2	Input buffer handle
Input	Binary(4)
Omissible Parameter Group:	
3	Low-level environment handle
Input	Binary(4)
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Put Command Buffer and Perform Get (QsnPutGetBuf) API sends the commands accumulated in a command buffer to the screen and performs a read operation. The command buffer must contain an input operation. If it has no input operation, a CPFA333 error occurs.

Authorities and Locks

None

Required Parameter Group

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer to be sent to the screen.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA308 E	Attempt to write data past end of display.
CPFA309 E	Invalid cursor position in command buffer.
CPFA30A E	Field length &1 not valid.
CPFA30B E	Invalid starting address for field.
CPFA30C E	Maximum allowable number of fields exceeded.
CPFA316 E	Saved data not valid.
CPFA31B E	From position &1, &2 greater than to position &3, &4.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA338 E	Command buffer contains an input operation.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve AID Code on Read (QsnRtvReadAID) API

Required Parameter:	
1	Input buffer handle
Input	Binary(4)
Omissible Parameter Group:	
2	AID code
Output	Char(1)
3	Error code
I/O	Char(*)
Returned Value:	
	AID code
Output	Char(1)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve AID Code on Read (QsnRtvReadAID) API determines the AID code corresponding to the input operation that filled the given input buffer.

Authorities and Locks

None

Required Parameter

Input buffer handle

INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation. The input buffer must be filled as a result of a Read Input Fields (QsnReadInp), Read Modified Fields (QsnReadMDT), or Read Modified Alternate (QsnReadMDTAlt) operation. If the input buffer is filled as a result of any other input operation, a CPFA32E error is issued.

Omissible Parameter Group

AID code

OUTPUT; CHAR(1)

The variable that contains the AID code when the QsnRtvReadAID API has completed. See AID-Generating Keys for a description of the possible values.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

AID code

OUTPUT; CHAR(1)

This API returns the value for the AID code parameter or X'00' if an error occurs during processing.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA32E E	Input data for query operation incorrect.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Available Data (QsnRtvAvailData) API

Required Parameter Group:

1 Input buffer handle

Input Binary(4)

2 Read command used

Output Binary(4)

Omissible Parameter Group:

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Available Data (QsnRtvAvailData) API is used to retrieve invited available data. If the invite active flag is on in a low-level environment description when an output operation is done, the end user is able to enter data. If a subsequent output operation is done, without checking for input using the QsnReadInvited API, and the user has entered data, DSM will store the end user's data in an internal input buffer and issue CPFA343. The Retrieve Available Data (QsnRtvAvailData) API will copy the data into the specified input buffer, as well as return the read command that is used to fill the buffer.

Authorities and Locks

None

Required Parameter Group

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer where available data should be moved.

Read command used
OUTPUT; CHAR(1)

The read command that was used to fill the input buffer. This can be used to determine what buffer manipulation API to use to retrieve the data from the input buffer. The possible values are:

X'42'	Read input fields
X'52'	Read MDT fields
X'72'	Read immediate
X'82'	Read MDT alternate

Omissible Parameter Group

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA320 E	Pointer parameter is null.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.

API introduced: V2R3

Retrieve Buffer Data Length (QsnRtvBufLen) API

Required Parameter Group:

1 Buffer handle

Input Binary(4)

Omissible Parameter Group:

2 Buffer data length

Output Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Buffer data length

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Buffer Data Length (QsnRtvBufLen) API returns the number of bytes of command data in a command buffer or of input data in an input buffer. After an indirect operation is applied to a command buffer, the QsnRtvBufLen API result reflects the increase in the underlying command stream to accommodate the command.

Authorities and Locks

None

Required Parameter

Buffer handle
INPUT; BINARY(4)

A handle for the buffer to be queried.

Omissible Parameter Group

Buffer data length
OUTPUT; BINARY(4)

The variable containing the buffer data length after the QsnRtvBufLen API has completed.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Buffer data length
OUTPUT; BINARY(4)

This API returns the value for the buffer data length parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Buffer Size (QsnRtvBufSiz) API

Required Parameter Group:

1 Buffer handle

Input Binary(4)

Omissible Parameter Group:

2 Buffer size

Output Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Buffer size

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Buffer Size (QsnRtvBufSiz) API returns the total number of bytes allocated for a command or input buffer. The result returned from this API is the current allocated buffer size, including any increments that may have taken place.

Authorities and Locks

None

Required Parameter

Buffer handle

INPUT; BINARY(4)

A handle for the buffer to be queried.

Omissible Parameter Group

Buffer size

OUTPUT; BINARY(4)

The variable containing the buffer size after the QsnRtvBufSiz API has completed.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Buffer size

OUTPUT; BINARY(4)

This API returns the value for the buffer size parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Cursor Address on Read (QsnRtvReadAdr) API

Required Parameter Group:	
1	Input buffer handle
Input	Binary(4)
Omissible Parameter Group:	
2	Cursor row
Output	Binary(4)
3	Cursor column
Output	Binary(4)
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Cursor Address on Read (QsnRtvReadAdr) API determines the row and column position of the cursor when the input operation that filled the given input buffer has completed. You must specify at least one of the cursor row or the cursor column parameter. If both of these parameters are omitted, a CPFA31E error occurs.

The input buffer must be filled as a result of a Read Input Fields (QsnReadInp), Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), Read Immediate (QsnReadImm), or Read Modified Immediate Alternate (QsnReadMDTImmAlt) operation. If the input buffer is filled as a result of any other input operation, a CPFA32E message is issued.

Authorities and Locks

None

Required Parameter

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation.

Omissible Parameter Group

Cursor row
OUTPUT; BINARY(4)

The variable that contains the row position of the cursor when the QsnRtvReadAdr API has completed.

Cursor column
OUTPUT; BINARY(4)

The variable that contains the column position of the cursor when the QsnRtvReadAdr API has completed.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA32E E	Input data for query operation incorrect.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Field Information (QsnRtvFldInf) API

Required Parameter Group:

1	Input buffer handle
Input	Binary(4)
2	Field number
Input	Binary(4)
3	Receiver variable
Output	Char(*)
4	Length of receiver variable
Input	Bin(4)

Omissible Parameter Group:

5	Low-level environment handle
Input	Binary(4)
6	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Field Information (QsnRtvFldInf) API retrieves information about a field in an input buffer filled by a Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), or Read Modified Immediate Alternate (QsnReadMDTImmAlt) operation.

To query the results from a Read Input Fields (QsnReadInp) or Read Immediate (QsnReadImm) operation, use the Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) and Retrieve Pointer to Field Data (QsnRtvFldDta) APIs. To query the result from any other input operations, use the Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) and Retrieve Pointer to Data in Input Buffer (QsnRtvDta) APIs.

Authorities and Locks

None

Required Parameter Group

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation.

Field number
INPUT; BINARY(4)

The number of the field to query, specified as n, where n is the nth field in the input buffer. The value specified must not be greater than the field count returned by the read operation.

Receiver variable
Output; CHAR(*)

The structure that will contain the result of the query when the QsnRtvFldInf API has completed.

Length of receiver variable
Input; BINARY(4)

The length of the receiver variable parameter.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Query Input Field Result

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(1)	Type of field
9	9	BINARY(4)	Row position of field
13	D	BINARY(4)	Column position of field
17	11	BINARY(4)	Length of data read
21	15	CHAR(11)	Reserved
32	20	PTR(SPP)	Pointer to field data

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Column position of field. The column position relative to the window of the specified field on the screen.

Length of data read. The length of the data read from the specified field.

Pointer to field data. A pointer to the data for the specified field.

Row position of field. The row position relative to the window of the specified field on the screen.

Type of field. The type of the specified field. The possible values are:

Value	Description
1	Normal field
2	Transparent field

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31A E	Incorrect field number value &1 specified.
CPFA31E E	Required parameter &1 omitted.
CPFA32E E	Input data for query operation incorrect.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API

Required Parameter Group:	
1	Input buffer handle
Input	Binary(4)
Omissible Parameter Group:	
2	Input data length
Output	Binary(4)
3	Error code
I/O	Char(*)
Returned Value:	
	Input data length
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API determines the number of bytes of input data contained in an input buffer after an input operation.

Authorities and Locks

None

Required Parameter

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation.

Omissible Parameter Group

Input data length
OUTPUT; BINARY(4)

The variable that contains the input data length when the QsnRtvDtaLen API has completed. This number may be smaller than the number of bytes actually read if the input buffer was not large enough to hold all the data. Use the Retrieve Number of Bytes Read from Screen (QsnRtvReadLen) API to determine the amount of data actually read from the screen. If the value returned by the QsnRtvReadLen API is less than the input data length, then truncation of the input data occurred.

Returned Value

Input data length
OUTPUT; BINARY(4)

This API returns the value for the input data length parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API

Required Parameter Group:

1 Input buffer handle

Input Binary(4)

Omissible Parameter Group:

2 Length of field data

Output Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Length of field data

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API determines the number of bytes of field data returned after a Read Input Fields (QsnReadInp) or Read Immediate (QsnReadImm) input operation. You can use the Retrieve Pointer to Field Data (QsnRtvFldDta) API to retrieve a pointer to this data so that you can parse the field values. Refer to the Read Input Fields (QsnReadInp) API for a description of the format of the data returned.

To query the results from a Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), or Read Modified Immediate Alternate (QsnReadMDTImmAlt) operation, use the Retrieve Number of Fields Read (QsnRtvFldCnt) and Retrieve Field Information (QsnRtvFldInf) APIs. To query the result from any other input operation, use the Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) and Retrieve Pointer to Data in Input Buffer (QsnRtvDta) APIs.

Authorities and Locks

None

Required Parameter

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation. The input buffer must be filled as a result of a QsnReadInp or QsnReadImm operation.

Omissible Parameter Group

Length of field data
OUTPUT; BINARY(4)

The variable that contains the field data length when the QsnRtvFldDtaLen API has completed. The field data length is 3 bytes less than the value returned by the QsnRtvDtaLen API. (The cursor and AID-key values account for the first 3 bytes of the input data returned).

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Length of field data
OUTPUT; BINARY(4)

This API returns the value for the length of field data parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA32E E	Input data for query operation incorrect.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Number of Bytes Read from Screen (QsnRtvReadLen) API

Required Parameter Group:	
1	Input buffer handle
Input	Binary(4)
Omissible Parameter Group	
2	Read data length
Output	Binary(4)
3	Error code
I/O	Char(*)
Returned Value:	
	Read data length
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Number of Bytes Read from Screen (QsnRtvReadLen) API returns the number of bytes of data read from the screen into an input buffer after an input operation.

Authorities and Locks

None

Required Parameter

Input buffer handle

INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation.

Omissible Parameter Group

Read data length

OUTPUT; BINARY(4)

The variable that contains the read data length when the QsnRtvReadLen API has completed. This number may be larger than the number of bytes actually contained in the buffer if the input buffer was not large enough to hold all the data. Use the Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API to determine the amount of data contained in the buffer or to determine if truncation occurred.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Read data length
OUTPUT; BINARY(4)

This API returns the value for the read data length parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA320 E	Pointer parameter is null.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Number of Fields Read (QsnRtvFldCnt) API

Required Parameter Group:

1 Input buffer handle

Input Binary(4)

Omissible Parameter Group:

2 Field count

Output Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Field count

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Number of Fields Read (QsnRtvFldCnt) API returns the number of fields contained in an input buffer after a Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), or Read Modified Immediate Alternate (QsnReadMDTImmAlt) operation. Use the Retrieve Field Information (QsnRtvFldInf) API to retrieve information about a specific field.

To query the results from a QsnReadInp or QsnReadImm operation, use the Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) and Retrieve Pointer to Field Data (QsnRtvFldDta) APIs. To query the result from any other input operation, use the Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) and Retrieve Pointer to Data in Input Buffer (QsnRtvDta) APIs.

Authorities and Locks

None

Required Parameter

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation. The input buffer must be filled as a result of a QsnReadMDT, QsnReadMDTAlt, or QsnReadMDTImmAlt operation.

Omissible Parameter Group

Field count
OUTPUT; BINARY(4)

The variable that contains the field count when the QsnRtvFldCnt API has completed.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Field count
OUTPUT; BINARY(4)

This API returns the value for the field count parameter, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA314 E	Memory allocation error.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA32E E	Input data for query operation incorrect.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Pointer to Data in Input Buffer (QsnRtvDta) API

Required Parameter Group:	
1	Input buffer handle
Input	Binary(4)
Omissible Parameter Group:	
2	Pointer to input data
Output	PTR(SPP)
3	Error code
I/O	Char(*)
Returned Value:	
	Pointer to input data
Output	PTR(SPP)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Pointer to Data in Input Buffer (QsnRtvDta) API returns a pointer to the first byte of input data in an input buffer after a read operation.

Authorities and Locks

None

Required Parameter

Input buffer handle

INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation.

Omissible Parameter Group

Pointer to input data

OUTPUT; PTR(SPP)

The variable that contains the pointer to the input data after the QsnRtvDta API has completed. You can use the Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API to retrieve the length of this data. Refer to the appropriate read operation for a description of the format of the data returned. The value returned by this API is equivalent to the data returned by the system on an input operation. This parameter must be on a 16-byte boundary.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Pointer to input data
OUTPUT; PTR(SPP)

This API returns the value for the pointer to input data parameter, or the null pointer otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1F E	Pointer is not on a 16 byte boundary.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Pointer to Field Data (QsnRtvFldDta) API

Required Parameter Group:

1 Input buffer handle

Input Binary(4)

Omissible Parameter Group:

2 Pointer to field data

Output PTR(SPP)

3 Error code

I/O Char(*)

Returned Value:

Pointer to field data

Output PTR(SPP)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Pointer to Field Data (QsnRtvFldDta) API returns a pointer to the first byte of field data in an input buffer after a Read Input Fields (QsnReadInp), Read Immediate (QsnReadImm), Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), or Read Modified Immediate Alternate (QsnReadMDTImmAlt) operation. You can use the Retrieve Length of Field Data in Buffer

(QsnRtvFldDtaLen) API to retrieve the length of this data. Refer to the "Read Input Fields (QsnReadInp) API" on page 100 for a description of the format of the data returned.

To query the results from a QsnReadMDT, QsnReadMDTAlt, or QsnReadMDTImmAlt operation, you can also use the QsnRtvFldCnt and QsnRtvFldInf APIs. To query the result from any other input operations, use the QsnRtvDtaLen and QsnRtvDta APIs.

Authorities and Locks

None

Required Parameter

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation. The input buffer must be filled as a result of a QsnReadInp or QsnReadImm operation.

Omissible Parameter Group

Pointer to field data
OUTPUT; PTR(SPP)

The variable that contains the pointer to the field data when the QsnRtvFldDta API has completed. The value returned by this API is the null pointer if the buffer contains no field data. Otherwise, it is equivalent to adding 3 bytes to the address returned by QsnRtvDta API. (The cursor and AID key values account for the first 3 bytes of input data returned.) This parameter must be on a 16-byte boundary.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Pointer to field data
OUTPUT; PTR(SPP)

This API returns the value for the pointer to field data parameter, or the null pointer otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPFA319 E	No data in input buffer.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA31E E	Required parameter &1 omitted.
CPFA32E E	Input data for query operation incorrect.
CPFA32F E	Buffer type incorrect.
CPF3C1F E	Pointer is not on a 16 byte boundary.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.

API introduced: V2R3

Retrieve Read Information (QsnRtvReadInf) API

Required Parameter Group:	
1	Input buffer handle
Input	Binary(4)
2	Query result
Output	Char(*)
3	Length of query result
Input	Binary(4)
Omissible Parameter Group:	
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Read Information (QsnRtvReadInf) API returns information about the input operation that filled the given input buffer.

Authorities and Locks

None

Required Parameter Group

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that contains the results of the input operation.

Query result
OUTPUT; CHAR(*)

The structure that contains the result of the query when the QsnRtvReadInf API has completed. The format of this structure is shown in "Format of the Query Result" on page 85.

Length of query result
INPUT; BINARY(4)

The length of the query result parameter. The minimum value must be 8.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Query Result

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(8)	Reserved
16	10	PTR(SPP)	Pointer to first byte of data
32	20	PTR(SPP)	Pointer to first byte of field data
48	30	BINARY(4)	Number of bytes of input data
52	34	BINARY(4)	Number of bytes of field data
56	38	BINARY(4)	Number of fields in input buffer
60	3C	BINARY(4)	Number of bytes of data received
64	40	BINARY(4)	Row location of cursor
68	44	BINARY(4)	Column location of cursor
72	48	CHAR(1)	AID code for AID-associated read request
73	49	CHAR(7)	Reserved

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

AID code for AID-associated read request. AID code corresponding to key pressed to service an AID-associated read request. The input buffer must be filled as a result of a QsnReadInp, QsnReadMDT, or QsnReadMDTAlt operation. If the input buffer is filled as a result of any other input operation, this field is set to X'00'. See AID-Generating Keys for a description of the possible values.

Column location of cursor. Column location of cursor when the input operation was serviced. The input buffer must be filled as a result of a QsnReadInp, QsnReadMDT, QsnReadMDTAlt, QsnReadImm, or QsnReadMDTImmAlt operation. If the input buffer is filled as a result of any other input operation, this field is set to -1.

Number of bytes of data sent. Number of bytes of data sent from screen. If this value is larger than the number of bytes of input data, then truncation occurs on the input operation.

Number of bytes of field data. Number of bytes of field data in input buffer. This does not include the 3 bytes of header information (the cursor row and column, and the AID byte). The input buffer must be filled as a result of a QsnReadInp, QsnReadMDT, QsnReadMDTAlt, QsnReadImm, or QsnReadMDTImmAlt operation. If the input buffer is filled as a result of any other input operation, this field is set to -1.

Number of bytes of input data. Number of bytes of input data in input buffer. This includes header information such as row and column position.

Number of fields in input buffer. Number of fields in input buffer. This does not include header information such as row and column position. The input buffer must be filled as a result of a QsnReadMDT, QsnReadMDTAlt, or QsnReadMDTImmAlt operation. If the input buffer is filled as a result of any other input operation, or the input data format cannot be determined, this field is set to -1.

Pointer to first byte of data. Pointer to first byte of data in input buffer. This includes header information such as row and column position.

Pointer to first byte of field data. Pointer to first byte of field data in input buffer. This will be the first byte of data following the header information (the cursor row and column, and the AID byte). If the buffer does not contain field data, this field is set to the null pointer.

Reserved. An ignored field.

Row location of cursor. Row location of cursor when the input operation was serviced. The input buffer must be filled as a result of a QsnReadInp, QsnReadMDT, QsnReadMDTAlt, QsnReadImm, or QsnReadMDTImmAlt operation. If the input buffer is filled as a result of any other input operation, this field is set to -1.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1F E	Pointer is not on a 16 byte boundary.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA319 E	No data in input buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA320 E	Pointer parameter is null.
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Screen Input APIs

The screen input APIs allow you to read data and other information from the screen. This includes field data, screen attributes, and the cursor address. The screen-input interfaces correspond, either directly or indirectly, to the 5250 data stream read commands.

The screen input APIs are:

- "Get AID (QsnGetAID) API" on page 88 (QsnGetAID) waits for an AID-generating key to be pressed.
- "Get Cursor Address (QsnGetCsrAdr) API" on page 89 (QsnGetCsrAdr) gets the current cursor address.
- "Get Cursor Address with AID (QsnGetCsrAdrAID) API" on page 91 (QsnGetCsrAdrAID) gets the current cursor address after an AID-generating key is pressed.
- "Put Input Command (QsnPutInpCmd) API" on page 93 (QsnPutInpCmd) issues a supplied read command.
- "Read from Invited Device (QsnReadInvited) API" on page 96 (QsnReadInvited) performs a read from invited device on a display that has already been invited.
- "Read Immediate (QsnReadImm) API" on page 98 (QsnReadImm) reads the contents of all input fields on the display without requiring an AID key to be pressed.
- "Read Input Fields (QsnReadInp) API" on page 100 (QsnReadInp) reads the contents of all input fields on the display requiring an AID key to be pressed.
- "Read Modified Alternate (QsnReadMDTAlt) API" on page 104 (QsnReadMDTAlt) reads the contents of all modified fields on the display, alternate form, requiring an AID key to be pressed.
- "Read Modified Fields (QsnReadMDT) API" on page 107 (QsnReadMDT) reads the contents of all modified fields requiring an AID key to be pressed.
- "Read Modified Immediate Alternate (QsnReadMDTImmAlt) API" on page 110 (QsnReadMDTImmAlt) reads the contents of all modified fields on the display, alternate form, without requiring an AID key to be pressed.
- "Read Screen (QsnReadScr) API" on page 112 (QsnReadScr) reads the contents of the screen without requiring an AID key to be pressed.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Get AID (QsnGetAID) API

Omissible Parameter Group:	
1	AID code
Output	Char(1)
2	Low-level environment handle
Input	Binary(4)
3	Error code
I/O	Char(*)
Returned Value	
	AID code
Output	Char(1)
Service Default Program: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Get AID (QsnGetAID) API waits for an AID-generating key to be pressed.

This command corresponds indirectly to the 5250 Read Input Fields command. Because the control characters specified on the underlying command are both X'00', this operation will cause the cursor to move to the insert cursor position when the keyboard is unlocked.

Authorities and Locks

None

Omissible Parameter Group

AID code
OUTPUT; CHAR(1)

The variable that contains the AID code when the QsnRtvReadAID API has completed.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

AID code
OUTPUT; CHAR(1)

This API returns the value for the AID code parameter, X'00' if a general error occurs during processing.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA326 E	Screen must be redrawn.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Get Cursor Address (QsnGetCsrAdr) API

Omissible Parameter Group:

1 Cursor row

Output Binary(4)

2 Cursor column

Output Binary(4)

3 Low-level environment handle

Input Binary(4)

4 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Get Cursor Address (QsnGetCsrAdr) API returns the current cursor address, without requiring an AID-generating key to be pressed. Either the cursor row or cursor column parameter must be specified. If both of these parameters are omitted, a CPFA31E error occurs.

This command corresponds indirectly to the 5250 Read Immediate command.

Authorities and Locks

None

Restrictions

The same restrictions apply as for the "Read Immediate (QsnReadImm) API" on page 98.

Omissible Parameter Group

Cursor row
OUTPUT; BINARY(4)

The variable that contains the cursor row when the QsnGetCsrAdr API has completed.

Cursor column
OUTPUT; BINARY(4)

The variable that contains the cursor column when the QsnGetCsrAdr API has completed.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Get Cursor Address with AID (QsnGetCsrAdrAID) API

Omissible Parameter Group:	
1	Cursor row
Output	Binary(4)
2	Cursor column
Output	Binary(4)
3	AID code
Output	Char(1)
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Public Default Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Get Cursor Address with AID (QsnGetCsrAdrAID API returns the cursor address after an AID-generating key is pressed. Either the cursor row or the cursor column parameter must be specified. If both of these parameters are omitted, a CPFA31E error occurs.

This command corresponds indirectly to the 5250 Read Input Fields command. Because the control characters specified on the underlying command are both X'00', this operation may cause the cursor to move to the default, or insert cursor, position when the keyboard is unlocked.

Authorities and Locks

None

Omissible Parameter Group

Cursor row
OUTPUT; BINARY(4)

The variable that contains the cursor row when the QsnGetCsrAdrAID API has completed.

Cursor column
OUTPUT; BINARY(4)

The variable that contains the cursor column when the QsnGetCsrAdrAID API has completed.

AID code
OUTPUT; CHAR(1)

The variable that contains the AID code when the QsnGetCsrAdrAID API has completed.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA326 E	Screen must be redrawn.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Put Input Command (QsnPutInpCmd) API

Required Parameter:

1 Command

Input Char(1)

Omissible Parameter Group:

2 Command data

Input Char(*)

3 Command data length

Input Binary(4)

4 Number of data bytes read

Output Binary(4)

5 Input buffer handle

Input Binary(4)

6 Command buffer handle

Input Binary(4)

7 Low-level environment handle

Input Binary(4)

8 Error code

I/O Char(*)

Returned Value:

Number of data bytes read

Output Binary(4)

Public Default Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Put Input Command (QsnPutInpCmd) API is used to issue data stream input commands that are not directly supported through a DSM API. An Escape (X'04') character is inserted in the stream directly before the command itself for both direct and indirect operations.

You must use this operation to issue an input command that is not directly supported by DSM as this will cause the appropriate underlying screen I/O operation to occur in order to retrieve input. You cannot, for example, use the Put Output Command (QsnPutOutCmd) API to issue an input command because no input data will be requested by the underlying DSM screen I/O operation.

The following command buffer handle and input buffer handle combinations are valid.

- The command buffer handle is specified with a nonzero value. The input buffer handle is omitted or specified with a zero value.

  This is an indirect operation. The command is stored in the command buffer without an I/O operation taking place.

- The command buffer handle is omitted or specified with a zero value. The input buffer handle is specified with a nonzero value.

  This is a direct operation. The input operation is issued to the screen, and the resulting input data is stored in the input buffer.
- Both a command buffer handle and an input buffer handle are specified with nonzero values.

  This is a direct operation. The input operation is appended to the command stream given by the command buffer, and the entire command stream is written to the display. The resulting input data is stored in the input buffer. The contents of the command buffer are not affected by this operation. That is, the input operation is not stored in the command buffer.

This operation corresponds to an Escape character followed by the specified command.

Authorities and Locks

None

Required Parameter

Command
INPUT; CHAR(1)

The 1-byte character code for the input command to be issued. For example, to issue a Save Partial Screen command, the command data should contain X'03' and the command data will contain the dimensions of the partial screen to be saved.

Omissible Parameter Group

Command data
INPUT; CHAR(*)

The data for the command to be issued.

Command data length
INPUT; BINARY(4)

The length of the command data parameter.

Number of data bytes read
OUTPUT; BINARY(4)

The variable that contains the number of data bytes returned after the QsnPutInpCmd API has completed if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Number of data bytes read
OUTPUT; BINARY(4)

This API returns the value for the number of data bytes read parameter if a direct operation was specified, or -1 if an error occurs during processing. If this is an indirect operation, this API returns zero if successful, or -1 if there was a general failure.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read from Invited Device (QsnReadInvited) API

Required Parameter Group:	
1	Input buffer handle
Input	Binary(4)
Omissible Parameter Group:	
2	Command buffer handle
Input	Binary(4)
3	Low-level environment handle
Input	Binary(4)
4	Return code
Output	Binary(4)
5	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	

The Read from Invited Device (QsnReadInvited) API issues a read from invited device operation. Data will be returned in the format corresponding to the read command used.

If the command buffer handle is specified and there is data to be sent in the command buffer, a QsnPutBuf will be issued to send the data to the screen. If no read command is in the command buffer, a read MDT command will be added to the data stream. Then the read from invited device will be issued.

The input buffer handle parameter must be specified.

See the appropriate read API for information on the format of the data returned.

Authorities and Locks

None

Restrictions

The invite active flag must be on in the low level environment description.

An error will be issued if the command buffer is empty, or not specified, and no other write has been done with the invite active flag on in the low level environment description.

Required Parameter Group

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations.

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to find the read command.

If no read command is found in the specified command buffer, a read MDT with null control characters will be added to the data stream. This is the equivalent of calling the QsnReadMDT API.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, -1 if there was a general failure, and -2 if the operation was a read from invited device which timed out.

Check the WAITRCD parameter on the display file specified in the low level environment description, to determine the time out value.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, -1 if there was a general failure, and -2 if the operation was a read from invited device which timed out.

Check the WAITRCD parameter on the display file specified in the low level environment description, to determine the time out value.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA309 E	Invalid cursor position in command buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA326 E	Screen must be redrawn.
CPFA327 E	Low level environment description value incorrect.

Message ID	Error Message Text
CPFA32F E	Buffer type incorrect.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Immediate (QsnReadImm) API

Omissible Parameter Group:

1	Number of field data bytes read
Output	Binary(4)
2	Input buffer handle
Input	Binary(4)
3	Command buffer handle
Input	Binary(4)
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)

Returned Value:

	Number of field data bytes read
Output	Binary(4)

Public Default Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Immediate (QsnReadImm) API reads the contents of all input fields on the display without requiring an AID key to be pressed. The command buffer handle or input buffer handle parameter must be specified as described in "Put Input Command (QsnPutInpCmd) API" on page 93.

The information returned depends on the condition of the master MDT bit. (See Modified Data Tag (MDT) Bit.) If the bit is not set, the input returned consists of the cursor address and an AID code only. If the bit is set, the input returned also includes the field data in the data portion of the input buffer. In each case, the returned cursor address indicates the current location of the cursor and an AID code of X'00'. The format of the field data returned is the same as that for the Read Input Fields (QsnReadInp) API.

This command corresponds directly to the 5250 Read Immediate command.

Authorities and Locks

None

Restrictions

This command must be the last command in the command buffer. A CPFA305 error is issued if there is a subsequent attempt to add another command to the specified command buffer after this command.

Omissible Parameter Group

Number of field data bytes read
OUTPUT; BINARY(4)

The variable that contains the number of field data bytes returned after the QsnReadImm API has completed if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations. See "Retrieve Pointer to Field Data (QsnRtvFldDta) API" on page 82 and "Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API" on page 76.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Number of field data bytes read
OUTPUT; BINARY(4)

This API returns the value for the number of field data bytes read parameter if a direct operation was specified, or -1 if an error occurs during processing. If this is an indirect operation, this API returns zero if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA302 E	Command buffer or input buffer parameters required.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.

Message ID	Error Message Text
CPFA305 E	Cannot add operation to command buffer.
CPFA309 E	Invalid cursor position in command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Input Fields (QsnReadInp) API

Required Parameter Group:

1 Control character byte 1

Input Char(1)

2 Control character byte 2

Input Char(1)

Omissible Parameter Group:

3 Number of field data bytes read

Output Binary(4)

4 Input buffer handle

Input Binary(4)

5 Command buffer handle

Input Binary(4)

6 Low-level environment handle

Input Binary(4)

7 Error code

I/O Char(*)

Returned Value:

Number of field data bytes read

Output Binary(4)

Public Default Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Input Fields (QsnReadInp) API reads the contents of all input fields on the display while requiring an AID-generating key to be pressed. The command buffer handle or input buffer handle parameter must be specified as described in "Put Input Command (QsnPutInpCmd) API" on page 93. The format of the data returned is:

Cursor Address in Row/Column FMT 2 bytes	AID Code 1 byte	Field Data	Field Data	Field Data

The information returned depends on the condition of the master MDT bit (see Modified Data Tag (MDT) Bit) and the AID-generating key pressed. If the bit is not set, the input returned consists of the cursor address and an AID code only. If the bit is set, the input returned also includes the field data in the data portion of the input buffer. In either case, the returned cursor address indicates the location of the cursor when the AID-generating key was pressed and the AID code for the AID-generating key the operator used. See AID-Generating Keys for a description of the AID-generating character values.

Field data is returned only when one of the following AID-generating keys is used:

Roll Up

Roll Down

Enter/Auto Record Advance

Auto Enter

An unmasked command function key

The field data, when returned, consists of the contents of all input fields as they appear on the display, unless resequencing has been specified. (See Resequencing.) Any attributes contained in a field are treated as data and returned as such. The attributes that start and end a field are not returned. These are considered to be outside the boundaries of the field. No field delimiters are added; data from each field is followed directly by the data from the next field. Data for nontransparent fields is formatted as follows:

- All nulls (leading, embedded, or trailing) are converted to blanks.
- If the specified field is a signed numeric field:
  - The last character of the field is stripped off.
  - The last location of the field is checked for a negative sign. If detected, the zone portion of the second to last character of the field is changed to a X'D'.

The data returned for transparent or CCSID-capable fields is not edited. A transparent field is defined with a transparency field control word X'84yy' (see "Set Field (QsnSetFld) API" on page 136.) The "Set Field with CCSID (QsnSetFldCC) API" on page 146 API creates CCSID-capable fields. Each field read is returned unedited with no intervening control information. If a field is both transparent and signed numeric, unpredictable results can occur in the field data. CDRA conversion may be peformed upon this data, see "Limitations and Restrictions" on page 292 for further details.

This command corresponds directly to the 5250 Read Input Fields command.

Restrictions

This command cannot be issued if the control unit supports ideographic data types. A CPFA306 error will occur if an attempt is made to issue this command to a control unit that supports ideographic data types.

Some control units, like those emulated by the Client Access program, do not support a control character associated with input commands. For such units, the control character specified would be ignored. A program could cause further actions to be suspended if, for example, the control character byte 2

specified to unlock the keyboard and this action was not specified elsewhere in the data stream. If the underlying control unit does not support a control character with input commands, you must specify the action to perform using the QsnWTD API.

Authorities and Locks

None

Required Parameter Group

Control character byte 1
INPUT; CHAR(1)

The operation for the display to perform after the read operation has been serviced. See Control Characters for a description of the control character values.

Control character byte 2
INPUT; CHAR(1)

The operation for the display to perform after the read operation and control character byte 1 have been serviced. See Control Characters for a description of the control character values.

Omissible Parameter Group

Number of field data bytes read
OUTPUT; BINARY(4)

The variable that contains the number of field data bytes returned after the QsnReadInp API has completed if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations. See "Retrieve Pointer to Field Data (QsnRtvFldDta) API" on page 82 and "Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API" on page 76.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Number of field data bytes read
OUTPUT; BINARY(4)

This API returns the value for the number of field data bytes read parameter if a direct operation was specified, or -1 if an error occurs during processing. If this is an indirect operation, this API returns zero if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA302 E	Command buffer or input buffer parameters required.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA309 E	Invalid cursor position in command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA31C E	Incorrect value for control character byte &1.
CPFA31E E	Required parameter &1 omitted.
CPFA326 E	Screen must be redrawn.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Modified Alternate (QsnReadMDTAlt) API

Required Parameter Group:

1	Control character byte 1
Input	Char(1)
2	Control character byte 1
Input	Char(1)

Omissible Parameter Group:

3	Field count
Output	Binary(4)
4	Input buffer handle
Input	Binary(4)
5	Command buffer handle
Input	Binary(4)
6	Low-level environment handle
Input	Binary(4)
7	Error code
I/O	Char(*)

Returned Value:

	Field count
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Modified Alternate (QsnReadMDTAlt) API reads the contents of all modified fields on the screen, alternate form, requiring an AID-generating key to be pressed. The QsnReadMDTAlt API is functionally equivalent to the QsnReadMDT API with the following exceptions:

- Leading and embedded nulls within the fields remain nulls. However, trailing nulls are stripped off.
- For fields consisting entirely of nulls, but with their MDT bit on, only the field's address is returned.

See "Read Modified Fields (QsnReadMDT) API" on page 107 for details.

This command corresponds directly to the 5250 Read MDT Alternate command.

Authorities and Locks

None

Restrictions

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

Some control units, like those emulated by the Client Access program, do not support a control character associated with input commands. For such units, the control character specified would be ignored. A program could cause further actions to be suspended if, for example, the control character byte 2 specified to unlock the keyboard and this action was not specified elsewhere in the data stream. If the underlying control unit does not support a control character with input commands, you must specify the action to perform using the QsnWTD API.

Required Parameter Group

Control character byte 1
INPUT; CHAR(1)

The operation for the display to perform after the read operation has been serviced. See Control Characters for a description of the control character values.

Control character byte 2
INPUT; CHAR(1)

The operation for the display to perform after the read operation and control character byte 1 have been serviced. See Control Characters for a description of the control character values.

Omissible Parameter Group

Field count
OUTPUT; BINARY(4)

The variable that will contain the number of input fields read after the QsnReadMDTAlt API has completed, if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations. See "Retrieve Pointer to Field Data (QsnRtvFldDta) API" on page 82 and "Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API" on page 76.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Field count
OUTPUT; BINARY(4)

Returns the value for the field count parameter if a direct operation was specified or -1 if an error occurs during processing. If this is an indirect operation, returns zero if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA302 E	Command buffer or input buffer parameters required.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA306 E	Command not supported by current device.
CPFA309 E	Invalid cursor position in command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA31C E	Incorrect value for control character byte &1.
CPFA31E E	Required parameter &1 omitted.
CPFA326 E	Screen must be redrawn.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Modified Fields (QsnReadMDT) API

Required Parameter Group:

1	Control character byte 1
Input	Char(1)
2	Control character byte 2
Input	Char(1)

Omissible Parameter Group:

3	Field count
Output	Binary(4)
4	Input buffer handle
Input	Binary(4)
5	Command buffer handle
Input	Binary(4)
6	Low-level environment handle
Input	Binary(4)
7	Error code
I/O	Char(*)

Returned Value:

	Field count
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Modified Fields (QsnReadMDT) API reads the contents of all modified fields on the screen requiring an AID-generating key to be pressed. The command buffer handle or input buffer handle parameter must be specified as described in "Put Input Command (QsnPutInpCmd) API" on page 93. See Control Characters for a description of the control character values. The format of the data returned is:

Cursor Row/Column	AID Code	SBA	Field Row/Column	Field Data	SBA	Field Row/Column	Field Data
2 bytes	1 byte	X'11'	2 bytes		X'11'	2 bytes	

The field data returned for transparent fields and CCSID-capable fields (created with "Set Field with CCSID (QsnSetFldCC) API" on page 146) includes an additional order and a length:

	Field Data	
Inbound Transparent Data order	Length of the following data	Transparent data or data in Field CCSID
X'10'	2 bytes	

The information returned depends on the state of the MDT bit for each field (see Modified Data Tag (MDT) Bit) and the AID-generating key used. If no bits are set, the input returned consists of the cursor

address and an AID code only. If at least one bit is set, the input may also include field data. In each case, the returned cursor address indicates the location of the cursor when the AID-generating key was pressed and the AID code for the AID-generating key the operator used.

Field data is returned only when one of the following AID-generating keys is used:

Roll Up

Roll Down

Enter/Auto Record Advance

An unmasked command function key

The field data, when returned, consists of the row and column address and the contents of each field that has an MDT bit on as they appear on the display, unless resequencing has been specified. (See Resequencing.) The input buffer query routines "Retrieve Number of Fields Read (QsnRtvFldCnt) API" on page 79 and "Retrieve Field Information (QsnRtvFldInf) API" on page 72 can be used to retrieve the value for each field. (To interpret the data directly, QsnRtvDta can be used to obtain a pointer for the data portion of the input buffer. The first data byte will be the SBA order for the first field as defined in the 5250 data stream documentation.)

Data for nontransparent fields is formatted as follows:

- Leading and embedded nulls within the field are translated to blanks, and trailing nulls are stripped off. Only the field's address is returned for fields whose MDT bit is on but which consist entirely of nulls.
- If the specified field is a signed numeric field:

  – The last character of the field is stripped off unless it was previously stripped because it was null.
  – The last location of the field is checked for a negative sign. If detected, the zone portion of the second to last character of the field is changed to a X'D'.

If the field is a transparent or CCSID-capable field, no formatting is performed.

If a field is both transparent and signed numeric, unpredictable results can occur in the field data. CDRA conversion may be peformed upon this data, see "Limitations and Restrictions" on page 292 for further details.

This command corresponds directly to the 5250 Read MDT Fields command.

Restrictions

Some control units, like those emulated by the Client Access program, do not support a control character associated with input commands. For such units, the control character specified would be ignored. A program could cause further actions to be suspended if, for example, the control character byte 2 specified to unlock the keyboard and this action was not specified elsewhere in the data stream. If the underlying control unit does not support a control character with input commands, you must specify the action to perform using the QsnWTD API.

Authorities and Locks

None

Required Parameter Group

Control character byte 1
INPUT; CHAR(1)

The operation for the display to perform after the read operation has been serviced. See Control Characters or a description of the control character values.

Control character byte 2
INPUT; CHAR(1)

The operation for the display to perform after the read operation and control character byte 1 have been serviced. See Control Characters for a description of the control character values.

Omissible Parameter Group

Field count
OUTPUT; BINARY(4)

The variable that contains the number of input fields read after the QsnReadMDT API has completed if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations. See "Retrieve Pointer to Field Data (QsnRtvFldDta) API" on page 82 and "Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API" on page 76.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Field count
OUTPUT; BINARY(4)

This API returns the value for the field count parameter if a direct operation was specified, or -1 if an error occurs during processing. If this is an indirect operation, this API returns zero if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.

Message ID	Error Message Text
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA302 E	Command buffer or input buffer parameters required.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA309 E	Invalid cursor position in command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA31C E	Incorrect value for control character byte &1.
CPFA31E E	Required parameter &1 omitted.
CPFA326 E	Screen must be redrawn.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Modified Immediate Alternate (QsnReadMDTImmAlt) API

Omissible Parameter Group:

1	Field count
Output	Binary(4)
2	Input buffer handle
Input	Binary(4)
3	Command buffer handle
Input	Binary(4)
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)

Returned Value:

	Field count
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Modified Immediate Alternate (QsnReadMDTImmAlt) API reads the contents of all modified fields on the display without requiring an AID-generating key to be pressed. Processing for this API is the same as for the Read Immediate (QsnReadImm) API, except that data is returned only for those fields that have the MDT bit on. The format of the data returned is the same as for the Read Modified Alternate (QsnReadMDTAlt) API.

See “Read Immediate (QsnReadImm) API” on page 98 and “Read Modified Alternate (QsnReadMDTAlt) API” on page 104 for details.

This command corresponds directly to the 5250 Read MDT Immediate Alternate command.

Restrictions

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

Authorities and Locks

None

Omissible Parameter Group

Field count
OUTPUT; BINARY(4)

The variable that contains the number of input fields read after the QsnReadMDTImmAlt API has completed if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations. See “Retrieve Pointer to Field Data (QsnRtvFldDta) API” on page 82 and “Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API” on page 76.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Field count
OUTPUT; BINARY(4)

This API returns the value for the field count parameter if a direct operation was specified, or -1 if an error occurs during processing. If this is an indirect operation, this API returns zero if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA302 E	Command buffer or input buffer parameters required.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA306 E	Command not supported by current device.
CPFA309 E	Invalid cursor position in command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Screen (QsnReadScr) API

Omissible Parameter Group:

1	Number of data bytes read
Output	Binary(4)
2	Input buffer handle
Input	Binary(4)
3	Command buffer handle
Input	Binary(4)
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)

Returned Value:

	Number of data bytes read
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Screen (QsnReadScr) API reads the contents of the entire screen without requiring an AID-generating key to be pressed. The command buffer handle or input buffer handle parameter must be specified as described in "Put Input Command (QsnPutInpCmd) API" on page 93.

The data returned consists of the contents of the entire display, including the attributes. No formatting or conversion is done. The data will be available in the data portion of the input buffer. The result of this

operation can be queried using the “Retrieve Length of Data in Input Buffer (QsnRtvDtaLen) API” on page 75 and the “Retrieve Pointer to Data in Input Buffer (QsnRtvDta) API” on page 81.

This command corresponds directly to the 5250 Read Screen command.

Authorities and Locks

None

Restrictions

The same restrictions apply as for the “Read Immediate (QsnReadImm) API” on page 98. In addition, this command cannot be issued if the control unit supports ideographic data types. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that supports ideographic data types.

Omissible Parameter Group

Number of data bytes read
OUTPUT; BINARY(4)

The variable that contains the number of data bytes returned after the QsnReadScr API has completed if a direct operation is specified. The parameter is not modified for an indirect operation and the value remains unchanged from whatever was passed.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer that receives the result of the input operation if a direct operation is specified. The result can be queried using the input buffer query operations. See “Retrieve Pointer to Field Data (QsnRtvFldDta) API” on page 82 and “Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API” on page 76.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Number of data bytes read
OUTPUT; BINARY(4)

This API returns the value for the number of data bytes read parameter if a direct operation was specified, or -1 if an error occurs during processing. If this is an indirect operation, this API returns zero if successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA302 E	Command buffer or input buffer parameters required.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA309 E	Invalid cursor position in command buffer.
CPFA313 E	Command buffer already contains an input operation.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Screen Output APIs

The screen output APIs are used to define fields and write data and other information to the screen.

The screen output interfaces are:

- "Delete Field ID Definition (QsnDltFldId) API" on page 115 (QsnDltFldId) deletes a field ID definition.
- "Generate a Beep (QsnBeep) API" on page 116 (QsnBeep) generates a beep.
- "Insert Cursor (QsnInsCsr) API" on page 118 (QsnInsCsr) sets the insert cursor address.
- "Pad between Two Screen Addresses (QsnWrtPadAdr) API" on page 121 (QsnWrtPadAdr) pads the screen with characters between two points.
- "Pad for N Positions (QsnWrtPad) API" on page 124 (QsnWrtPad) pads the screen for a specified number of characters.
- "Put Output Command (QsnPutOutCmd) API" on page 127 (QsnPutOutCmd) writes a data stream command.
- "Set Cursor Address (QsnSetCsrAdr) API" on page 129 (QsnSetCsrAdr) sets the position of the cursor on the screen.
- "Set Error State (QsnSetErr) API" on page 132 (QsnSetErr) places the keyboard into prehelp error state and optionally places a string on the error line with cursor positioning support.
- "Set Field (QsnSetFld) API" on page 136 (QsnSetFld) defines an input field on the screen at a given row and column.
- "Set Field with CCSID (QsnSetFldCC) API" on page 146 (QsnSetFldCC) defines a CCSID-capable input field on the screen at the given row and column.
- "Set Output Address (QsnSetOutAdr) API" on page 152 (QsnSetOutAdr) sets the current display address.
- "Write Data (QsnWrtDta) API" on page 155 (QsnWrtDta) writes data to the display at a given row and column with standard attributes.
- "Write Data with CCSID (QsnWrtDtaCC) API" on page 159 (QsnWrtDtaCC) writes data to the display at a given row and column using standard attributes.
- "Write Structured Field Major (QsnWrtSFMaj) API" on page 163 (QsnWrtSFMaj) writes the major structure of a structured field.
- "Write Structured Field Minor (QsnWrtSFMin) API" on page 166 (QsnWrtSFMin) writes the minor structure of a structured field.
- "Write to Display (QsnWTD) API" on page 169 (QsnWTD) issues a Write to Display command.

- "Write Transparent Data (QsnWrtTDta) API" on page 172 (QsnWrtTDta) writes transparent data to the display at a given row and column.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Delete Field ID Definition (QsnDltFldId) API

Required Parameter:

1 Field ID

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Delete Field ID Definition (QsnDltFldId) API deletes a field ID definition. The screen appearance, including the fields defined on the screen, are not affected by this command.

Authorities and Locks

None

Required Parameter

Field ID

INPUT; BINARY(4)

The ID for the field definition to be deleted. Subsequent references to this field ID result in a CPFA33C error. This parameter must be specified with a nonzero valid field value.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA33C E	Undefined field ID &1.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Generate a Beep (QsnBeep) API

Omissible Parameter Group:

1	Command buffer handle
Input	Binary(4)
2	Low-level environment handle
Input	Binary(4)
3	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Generate a Beep (QsnBeep) API generates a beep. The display address is not affected by this command.

This command corresponds directly to the 5250 Write to Display (WTD) command with control character 1 equal to X'00' and control character 2 equal to X'04'. If this is an indirect operation, this API issues a new WTD command to the command buffer.

Authorities and Locks

None

Restrictions

The same restrictions apply as for the Write Data (QsnWrtDta) API (see "Restrictions" on page 156).

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the beep is generated immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API Introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Insert Cursor (QsnInsCsr) API

Omissible Parameter Group:	
1	Field ID
Input	Binary(4)
2	Cursor row
Input	Binary(4)
3	Cursor column
Input	Binary(4)
4	Command buffer handle
Input	Binary(4)
5	Low-level environment handle
Input	Binary(4)
6	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Insert Cursor (QsnInsCsr) API sets the insert-cursor address. The insert-cursor address specifies the home-cursor address. (The position of the cursor when the host system unlocks the keyboard and the display station operator presses the Home key.) The display address is not affected by this command if this is an indirect operation and the target command buffer contains an active Write to Display (WTD) command. Otherwise, a WTD command is inserted that resets the display address to row 1 column 1.

If bit 1 of the associated Write to Display command is set to 0 (which is the default), the cursor will be moved on the screen when the QsnInsCsr API is called. To prevent the cursor from being moved, the control character byte 2 bit should be set to 1 and the Write to Display (QsnWTD) operation should be explicitly issued to a command buffer used by the QsnInsCsr API.

This command corresponds indirectly to the 5250 Write to Display (WTD) command with an Insert Cursor order. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Authorities and Locks

None

Restrictions

The same restrictions apply as for the Write Data (QsnWrtDta) API.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. Either the field ID or the row and column parameters must be specified.

Cursor row
INPUT; BINARY(4)

The row at which to position the insert cursor. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

Cursor column
INPUT; BINARY(4)

The column at which to position the insert cursor. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the insert cursor is positioned at the specified location immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA33C E	Undefined field ID &1.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Pad between Two Screen Addresses (QsnWrtPadAdr) API

Required Parameter Group:	
1	Pad character
Input	Char(1)
2	To row
Input	Binary(4)
3	To column
Input	Binary(4)
Omissible Parameter Group:	
4	From row
Input	Binary(4)
5	From column
Input	Binary(4)
6	Command buffer handle
Input	Binary(4)
7	Low-level environment handle
Input	Binary(4)
8	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Pad between Two Screen Addresses (QsnWrtPadAdr) API pads the display repeatedly with a selected character between two positions on the screen. The current display address is set to the position given by the to-row and to-column values plus one. Padding may occur outside the logical window area defined by the low-level environment window mode setting.

This command corresponds indirectly to the 5250 Write to Display (WTD) command with a Set Buffer Address order (if the from row and from column parameters are specified) and a Repeat to Address order. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Authorities and Locks

None

Restrictions

The same restrictions apply as for the Write Data (QsnWrtDta) API.

Required Parameter Group

Pad character

INPUT; CHAR(1)

The character to pad the screen with.

To row

INPUT; BINARY(4)

The row at which to write the last pad character. If the position to pad to is less than the position to pad from, a CPFA31B error is issued. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

To column

INPUT; BINARY(4)

The column at which to write the last pad character. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Omissible Parameter Group

From row

INPUT; BINARY(4)

The row at which to write the first pad character. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the from-row and from-column parameters are omitted, the pad characters are written starting at the current display address. If the command is a direct operation or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1, prior to writing the pad characters. Both the from-row and from-column parameters must be specified, or both parameters must be omitted.

From column

INPUT; BINARY(4)

The column at which to write the first pad character. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if

offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the screen is padded with the character specified between the positions specified inclusively. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA31B E	From position &1, &2 greater than to position &3, &4.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Pad for N Positions (QsnWrtPad) API

Required Parameter Group:

1	Pad character
Input	Char(1)
2	Number of bytes
Input	Binary(4)

Omissible Parameter Group:

3	Field ID
Input	Binary(4)
4	From row
Input	Binary(4)
5	From column
Input	Binary(4)
6	Command buffer handle
Input	Binary(4)
7	Low-level environment handle
Input	Binary(4)
8	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Pad for N Positions (QsnWrtPad) API pads the display with the given pad character for the specified number of bytes. Padding starts at the row and column specified, or at the current display address if these parameters are omitted. To allow the QsnWrtPad operation to insert a group of characters without overwriting the ending screen attribute, the following conditions must be satisfied:

- The operation is an indirect operation.
- The row and column parameters are omitted.
- The low-level environment description does not indicate that DBCS data is being used.
- The previous command saved in the command buffer was a Write Data (QsnWrtDta) operation.

If the row and column parameters are omitted and the command is either a direct operation or the buffer specified does not contain a preceding output operation that sets the display address, then the current display address is set to row 1, column 1, prior to writing the pad characters.

If the from-row and from-column parameters are specified, this command corresponds indirectly to the 5250 Write to Display (WTD) command with a Set Buffer Address order. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in the buffer.)

Authorities and Locks

None

Restrictions

The same restrictions apply as for the Write Data (QsnWrtDta) API.

Required Parameter Group

Pad character
INPUT; CHAR(1)

The pad character to pad the screen with.

Number of bytes
INPUT; BINARY(4)

The number of bytes to pad the screen for.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. If neither the field ID or row and column parameters are specified, the current display address is used.

From row
INPUT; BINARY(4)

The row at which to write the first pad character. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the from-row and from-column parameters are omitted, the pad characters are written starting at the current display address. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1, prior to writing the pad characters. Both the from-row and from-column parameters must be specified, or both parameters must be omitted.

From column
INPUT; BINARY(4)

The column at which to write the first pad character. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full

screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the pad characters are written to the screen at the current display address. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Put Output Command (QsnPutOutCmd) API

Required Parameter Group:	
1	Command
Input	Char(1)
Omissible Parameter Group:	
2	Command data
Input	Char(*)
3	Command Data Length
Input	Binary(4)
4	Command buffer handle
Input	Binary(4)
5	Low-level environment handle
Input	Binary(4)
6	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Put Output Command (QsnPutOutCmd) API is used to issue data stream commands that are not directly supported through a DSM API. An escape (X'04') character is inserted in the stream directly before the command itself for both direct and indirect operations.

Note: The Write Data (QsnWrtDta) API should be used for issuing Write to Display command orders such as the Write Extended Attributes order.

This operation corresponds to an escape character followed by the specified command.

Authorities and Locks

None

Required Parameter

Command
INPUT; CHAR(1)

The 1-byte character code for the output command to be issued. For example, to issue a Restore Partial Screen, the command data should contain X'13', the command data will contain the restore data length followed by the restore data, and the command data length will be 2 plus the restore data length.

Omissible Parameter Group

Command data
INPUT; CHAR(*)

The data for the command to be issued.

Command data length
INPUT; BINARY(4)

The length of the command data parameter. If 0 is specified, the command data parameter is ignored. Otherwise, the command data parameter is required.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the command is sent to the display. If the command being sent is an input command, you must specify a command buffer and then use the Put Command Buffer and Perform Get (QsnPutGetBuf) API to issue the command and retrieve the resulting input. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.

Message ID	Error Message Text
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Set Cursor Address (QsnSetCsrAdr) API

Omissible Parameter Group:

1	Field ID
Input	Binary(4)
2	Cursor row
Input	Binary(4)
3	Cursor column
Input	Binary(4)
4	Command buffer handle
Input	Binary(4)
5	Low-level environment handle
Input	Binary(4)
6	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Set Cursor Address (QsnSetCsrAdr) API sets the position of the cursor on the screen. The display address is not affected by this command if this is an indirect operation and the target command buffer contains an active Write to Display (WTD) command. Otherwise, a WTD command will be inserted that resets the display address to row 1 column 1.

This command corresponds indirectly to the 5250 Write to Display command (WTD) with an Insert Cursor or Move Cursor order. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.) The Move Cursor order is used if the control unit supports it (based on the 5250 Query command). Otherwise, the Insert Cursor order is used.

If the Move Cursor order is supported, this API sets the cursor without modifying the home address and without regard to the state of the keyboard. Otherwise, the behavior is the same as that of the Insert Cursor (QsnInsCsr) API.

If multiple QsnSetCsrAdr operations are applied to the same command buffer, only the last QsnSetCsrAdr operation is in effect. The last QsnSetCsrAdr or QsnInsCsr operation determines the cursor position. If the Move Cursor order is used, the QsnSetCsrAdr negates any previous QsnInsCsr commands in the command buffer, except for the last QsnInsCsr, which sets the home position. If the Insert Cursor order is used, it negates any previous QsnInsCsr operations. If the Move Cursor order is supported, you can set the home position and then move the cursor by issuing a QsnInsCsr first, followed by a QsnSetCsrAdr operation.

Authorities and Locks

None

Restrictions

The same restrictions apply as for the Write Data (QsnWrtDta) API.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. Either the field ID or the row and column parameters must be specified.

Cursor row
INPUT; BINARY(4)

The row at which to position the cursor. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

Cursor column
INPUT; BINARY(4)

The column at which to position the cursor. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the cursor is positioned at the specified location immediately. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA33C E	Undefined field ID &1.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Set Error State (QsnSetErr) API

Omissible Parameter Group:

1	Message
Input	Char(*)
2	Message length
Input	Binary(4)
3	Field ID
Input	Binary(4)
4	Cursor row
Input	Binary(4)
5	Cursor column
Input	Binary(4)
6	Starting monochrome attribute
Input	Char(1)
7	Ending monochrome attribute
Input	Char(1)
8	Starting color attribute
Input	Char(1)
9	Ending color attribute
Input	Char(1)
10	Command buffer handle
Input	Binary(4)
11	Low-level environment handle
Input	Binary(4)
12	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Set Error State (QsnSetErr) API places the keyboard into prehelp error state and optionally places a string on the error line. To place the keyboard in the prehelp error state, you must follow this API with an AID-associated read API such as QsnReadInp.

Either the cursor or the message parameters must be specified to make the command valid. If neither of these are used, a CPFA305 error is issued. If a cursor position is specified, the cursor is moved immediately to the location given. This does not affect the cursor address set by the Insert Cursor (QsnInsCsr) API.

When the operator presses the Help key (prehelp error state only) in response to the error condition, the message No help text is available is displayed.

This command corresponds directly to the 5250 Write Error Code command.

Authorities and Locks

None

Omissible Parameter Group

Message

INPUT; CHAR(*)

The message to be displayed. This parameter is required if the message length parameter is specified as a nonzero value. The message data, including the screen attributes, must not exceed 132 characters for devices that are in 27x132 mode, or 80 characters for all other devices. A CPFA310 error is issued if the message data is too long.

Message length

INPUT; BINARY(4)

The number of bytes of message data to be displayed.

Field ID

INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address.

Cursor row

INPUT; BINARY(4)

The row at which to position the cursor when the message is displayed. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the field ID and the row and column parameters are omitted, the cursor is not moved. The row and column parameters must be specified together, or both parameters must be omitted.

Cursor column

INPUT; BINARY(4)

The column at which to position the cursor when the message is displayed. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Starting monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. If this parameter is omitted or specified as X'00', a starting attribute of high intensity blink is inserted. See Screen Attribute Characters for a description of the screen attribute values. The starting attribute is selected as for the QsnWrtDta API.

Ending monochrome attribute
INPUT; CHAR(1)

The ending screen attribute for monochrome displays. If this parameter is omitted or specified as X'00', an ending attribute of nondisplay is inserted. The ending attribute is selected as for the QsnWrtDta API.

Starting color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no initial attribute is written to the display for the data.

Ending color attribute
INPUT; CHAR(1)

The ending screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no ending attribute is written to the display for the data.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the error state is entered, the cursor is moved to the specified position, and the message, if specified, is displayed. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.

Message ID	Error Message Text
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA30D E	Invalid screen attribute.
CPFA30F E	Required parameter not specified.
CPFA310 E	Error message data/screen attributes exceed display width.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA33F E	Error occurred during data conversion.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Set Field (QsnSetFld) API

Omissible Parameter Group:

1	Field ID
Input	Binary(4)
2	Field length
Input	Binary(4)
3	Row
Input	Binary(4)
4	Column
Input	Binary(4)
5	Field format word (FFW)
Input	Char(2)
6	Field control words (FCW)
Input	Char(*)
7	Number of field control words
Input	Binary(4)
8	Monochrome attribute
Input	Char(1)
9	Color attribute
Input	Char(1)
10	Command buffer handle
Input	Binary(4)
11	Low-level environment handle
Input	Binary(4)
12	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Set Field (QsnSetFld) API defines an input field on the screen at the given row and column. The following occurs when this command is issued to the control unit as a direct operation or when the buffer containing the command is written out:

- Any outstanding AID requests are cleared.
- The keyboard is locked.

- If there is an entry in the format table whose starting address is equal to the address for this field, then that entry is modified. The FFW of the existing entry is replaced by the new FFW and the previous screen starting attribute is overlaid with the new screen starting attribute. The ending screen attribute is not rewritten. All FCWs and the length parameter are ignored. See the 5250 data stream documentation for details.
- If no entry can be found in the table for the field being defined, a new entry will be added to the end of the table. However, the address must be greater than the ending address of the field currently defined last in the format table or an error will occur. If the new entry is valid, it will contain the field's FFW, the optional FCWs, and the field's starting and ending address. An error will occur if an attempt is made to define too many fields on the screen (see the 5250 data stream documentation for details).

The display address after this operation will be the starting field address minus 1 if row and column are specified as valid positive integers or if this is the first field specified within the current WTD command. Otherwise, the display address will be one position past the ending screen attribute.

This command corresponds indirectly to the 5250 Write to Display (WTD) command with a Set Buffer Address order and a Start of Field order if the row and column parameters are specified. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Authorities and Locks

None

Restrictions

The same restrictions apply as for the Write Data (QsnWrtDta) API, with the exception that the trailing field attribute can be written past the end of the screen. (It will be suppressed by the control unit.)

Omissible Parameter Group

Field ID

INPUT; BINARY(4)

The field ID to be associated with this field. The value specified can be any nonzero integer value. For APIs that accept a field ID parameter, this value can be subsequently used instead of the row and column parameter to specify a screen address. If the given ID is already defined, this operation will redefine that field ID with the values specified. To remove a field ID definition, use the QsnDltFldId API.

If a previously defined field ID is supplied and some or all of the parameters are omitted, the field is defined using the current field definition values for those omitted parameters.

If this field is omitted or specified with a value of zero, then no field ID is associated with this field description.

Field length

INPUT; BINARY(4)

The length of the field being defined. If no field ID is specified, the length must be a positive integer value greater than 1 for signed numeric fields and greater than 0 for all other field types. The entire field must fit on the display. If a field ID is specified with a nonzero value, the length may be 0, in which case a field will not be defined on the screen; however, this will associate the field definition with the specified field ID.

Row INPUT; BINARY(4)

The row at which to define the field. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row

location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

The starting field address will be the row and column locations given if both parameters are specified. Otherwise, it will be the current display address plus 1. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1, prior to writing the initial screen attribute and the field definition. The ending field address for this field is the starting field address plus the field length.

If a field ID is supplied along with a row and column, the row and column parameters will be stored as specified. These parameters will be used as relative or actual screen positions on a subsequent operation, depending upon the window mode setting for the environment supplied with that operation.

If a previously undefined field ID is supplied with this operation, the row and column parameters must be specified. Also, the row and column parameters must both be specified or omitted; one cannot be specified if the other is omitted. A CPFA307 error occurs if an incorrect cursor position is specified. On some devices, row and column can both be specified as 1, which will cause the field to be defined at row 1, column 1, with a screen attribute of normal (X'20'). If this is the case, then any initial screen attribute parameters specified are ignored. This is only supported by certain devices. Whether or not this is supported can be determined by the Query 5250 (QsnQry5250) API.

Column

INPUT; BINARY(4)

The column at which to write the data. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Field format word (FFW)

INPUT; CHAR(2)

The field format word is a 2-byte value that controls the type of the field being defined. Table 6 (page 140) shows the field types, and the corresponding bit to be set for each type. To omit this parameter, specify X'00' in both characters of the parameter. You must specify this parameter to define an input field, and it is required if a field control word is specified.

Field control words (FCW)

INPUT; CHAR(*)

An array of 2-byte field control words. The field control words are 2-byte values that request certain functions to be performed. Table 7 (page 143) shows the valid field control word values, their function, and mnemonics for those values.

The 5250 CCSID-based I/O specific FCWs are not allowed ("Set Field with CCSID (QsnSetFldCC) API" on page 146 must be used to create CCSID-capable fields). If the CCSID-based I/O CCSID or Maximum data length FCWs are given here, a CPFA332 will result. FCWs will not be exhaustivly checked to see if they are formatted correctly or to see if the function requested is valid for the current device. However, some FCWs are checked against the support provided by the device and a CPFA306 signaled if an incompatibility is found. Table 8 (page 144) shows the display capablility and FCW combinations that are valid.

Errors not found here may be detected and reported when the FCW is required during subsequent command and keystroke processing. See the 5250 data stream documentation for further details about the meaning and use of these functions.

Number of field control words
INPUT; BINARY(4)

The number of control words in the field control word array. Omitting this parameter or specifying it with a value of 0 indicates that no field control words are specified with the FCW parameter. If this parameter is specified with a nonzero value, the FCW parameter is required; if the FCW parameter is omitted, a CPFA31E error is issued.

Monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. A screen attribute is required for defining a field on the screen; if this parameter is omitted and monochrome attributes are to be used, X'20' is assumed. The initial screen attribute is written one position to the left of the starting field address. The ending screen attribute (X'20') is supplied by the controller and written at the end-of-field address plus 1.

The monochrome attribute and color attribute parameters consist of 1 byte that will be used as the screen attribute for a monochrome or a color display, respectively. One of these parameters will be selected based on the underlying display type, and the other will be discarded. See Screen Attribute Characters for a description of the screen attribute values.

Color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. A screen attribute is required for defining a field on the screen; if this parameter is omitted and color attributes are to be used, X'20' is assumed. See Screen Attribute Characters for a description of the screen attribute values.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the starting field address will be the supplied location. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Field Format Word

The following table shows the field types and the bits to set for each type for the field format word (FFW).

Table 6. Field Format Words

Field Type	Bit To Set	Mnemonic	Description
	0-15	QSN_NO_FFW	If bits 0 to 15 are set to 0, then no FFW is used. The field will be defined as an output field and its contents will not be returned by operations such as the Read Input Fields (QsnReadInp) API.
	0-1		The first two bits of a field format word must be 01. The mnemonic for every other field type includes this setting.
Bypass	2	QSN_FFW_BYPASS	If this bit is set, this is a bypass field and entries are not allowed in it. If the operator tries to enter something in this field, an error results.
Dup Enable	3	QSN_FFW_DUP	If this bit is set, duplication is allowed in the field. The controller repeats X'1C' from the cursor position to the end of the field when the operator presses the Dup key; this shows on the display as an overstruck asterisk.
Modified data tag (MDT)	4	QSN_FFW_MDT	Setting this bit sets the modified data tag for this field. The field will then be read with the Read Modified Fields (QsnReadMDT) API (see "Read Modified Fields (QsnReadMDT) API" on page 107) as if the operator had modified it.
Field Shift/ Edit Specification	**Bits 5-7**	**Mnemonic**	**Description**
Alphabetic shift	000	QSN_FFW_ALPHA_SHIFT	The field accepts all characters. The shift keys are acknowledged. The characters on the lower symbol of each key are valid.
Alphabetic only	001	QSN_FFW_ALPHA_ONLY	The field accepts only characters A-Z (both uppercase and lowercase) plus the comma (,), period (.), minus (-), space, and DUP (if the DUP-enable bit is on in the associated Field Format Word (FFW)). Other characters cause operator errors. Some special characters are also acceptable (see the 5250 data stream documentation).
Numeric shift	010	QSN_FFW_NUM_SHIFT	The field accepts all characters from all keys.
Numeric only	011	QSN_FFW_NUM_ONLY	The field accepts only characters 0-9 and the comma (,), period (.), minus (-), plus (+), space, and DUP (if the DUP-enable bit is on in the associated Field Format Word (FFW)). Other characters cause operator errors. The unit position of this field will carry the sign digit for the field. If the field is exited with the Field - key, the last character in the field will be 'D' zoned, unless the last character in the field is a '+', '-', ',', '.', or space, in which case an error will be posted. In a center-adjusted field, the field will be center-adjusted before any 'D' zoning or testing of the sign character is performed. When a negative field (from the Field - key) is returned, the units digit will have a 'D' zone.
Katakana shift	100	QSN_FFW_KATA	This is the same as the alphabetic shift except that the keyboard is placed in the Katakana shift on the Japan Katakana data entry, typewriter, and G keyboards. This reverses the order of the cursor direction with respect to the screen. If the display is in bidirectional mode, this changes the cursor direction to left to center; otherwise, it changes the cursor direction to center to left.
Digits only	101	QSN_FFW_DIGIT_ONLY	The field allows keys 0-9 and DUP (if the DUP-enable bit is on in the associated Field Format Word (FFW)).

Field Type	Bit To Set	Mnemonic	Description
I/O	110	QSN_FFW_IO	This field will not accept any data keys from the keyboard. An operator error is posted if keystrokes are entered in this field. The operator may move the cursor into and out of this field similar to operation in any non-bypass input field (that is, Field Advance will position the cursor to the start of the field). This field can be used for input from feature devices such as a magnetic stripe reader of selector light pen while data input from the keyboard is excluded. The Field +, Field Exit, and Dup keys are valid for this field and performance is the same as that for any non-bypass input field.
Signed Numeric	111	QSN_FFW_SIGNED_NUMERIC	The field allows keys 0-9 and DUP (if the DUP-enable bit is on in the associated Field Format Word (FFW)). Typing any other character will cause an operator error display. This field reserves the center-hand position for a sign display (- for negative and null for positive); therefore, the largest number of characters that can be entered into this field is one less than the field length. A signed numeric field less than 2 characters long will cause an error to be flagged. No digit may be keyed into the centermost position; however, the cursor can be positioned there by using the cursor movement keys and then followed by the F+ or F- key. This allows changing the sign without affecting the rest of the field.
Signed Numeric (continued)			If this field is not a mandatory fill or center-adjust field, it is still handled as if it were specified as a center-adjust blank fill field. If the Field - key is used to exit this field, the field will be center-adjusted and a negative sign placed in the centermost position of the field. The Field Exit or Field + key will insert a blank in the centermost position and center-adjust this field. Before this field is returned on an input operation, it is changed as follows: • If the centermost character is a negative sign, the zone of the low order digit is set to a X'D'. • If the centermost character is not a negative sign, the low order digit is not changed. In either case, the centermost sign position is not sent to the host.
Field Type	**Bit to Set**	**Mnemonic**	**Description**
Auto Enter	8	QSN_FFW_AUTO_ENTER	If this bit is set, this is an auto enter field. The auto enter function occurs only for valid Field Exit/Field +, Field -, and Dup exit keys, or if the last data character position is typed into the auto enter field. After any required center-adjust, mandatory fill, data duplicating, or check digit operations are successfully performed, the auto enter function causes the panel to be sent to the host as if the Enter key had been pressed.
Field Exit Required	9	QSN_FFW_AUTO_FER	If this bit is set, a nondata key must be typed to leave the field. When the last data character is typed into the last position of the field, the cursor will remain under the character and blink, signifying the controller is waiting for an exit key. Any nondata key will satisfy the exit requirement (including cursor movement or function keys).

Field Type	Bit To Set	Mnemonic	Description
Monocase	10	QSN_FFW_AUTO_MONOCASE	If this bit is set, then regardless of the shift state of the typewriter keyboard, only the uppercase A-Z is entered into the field being typed (that is, if a lowercase a is typed, the uppercase A is entered). All other characters are unaffected. Certain characters on some typewriter keyboards also will be translated to uppercase (see the 5250 data stream documentation).
Reserved	11		
Mandatory Enter	12	QSN_FFW_ME	If this bit is set, the operator must enter something in the field before the controller allows the Enter key to be active. The controller recognizes the state of these fields by checking the MDT bit for the field. If the operator tries to bypass the field using a Field +, Field -, or Field Exit key, an error occurs.
Center-Adjust/ Mandatory Fill	**Bits 13-15**	**Mnemonic**	**Description**
No adjust specified	000	QSN_FFW_NOADJUST	No field adjustment occurs.
Reserved	001		
Reserved	010		
Reserved	011		
Reserved	100		
Center-adjust, zero fill	101	QSN_FFW_RA_ZERO	All leftmost unoccupied positions of a field are filled with zero. Characters are center-adjusted and spaces are zero-filled. The fill character will appear on the display. center-adjust is only activated by keying the Field Exit, Field +, or Field - keys. The field is center-adjusted from the first non-null character to the left of the cursor when one of these keys is depressed. If a center-adjust field is left through cursor movement keys, the field will remain as is (not center-adjusted). center-adjust fields longer than 15 characters might cause a slow response that would result in a keyboard overrun. A center-adjust specified field has an implied field exit required function. The Dup key will fill a center-adjust field from the cursor to the end of the field with the Dup character (X'1C'), but the field will not be center-adjusted. After typing the first character into a center adjust field, but prior to exiting the field (using cursor movement or exit keys), the Enter key will cause an operator error to be posted; that is, Enter is invalid from an active center-adjust field.
Center-adjust, blank fill	110	QSN_FFW_RA_BLANK	The field behavior is the same as for center-adjust, zero fill, but the fill character is blank instead of zero.

Field Type	Bit To Set	Mnemonic	Description
Mandatory fill	111	QSN_FFW_MF	Once any data has been entered into the field, the field must be completely filled before exiting it. Any attempt to leave an unfilled field causes an error. The cursor may pass into and out of a mandatory fill field as a result of cursor movement keys without fill-checking or error posting by the controller, as long as no data is entered into the field. The Dup key will fill the field from the cursor to the end of field with the Dup characters X'2C', and then the entire field will be checked for any nulls (an error is posted if a null is found). A mandatory fill field with nulls can be returned under the following conditions: 1. The field was initialized with nulls and with the MDT bit on. 2. The Erase Input, Field Exit, or Field + key is used from the first position of the field. The field is filled with nulls and the MDT bit is set on. The above fields, with no further entry, can be returned with all data as blanks on a Read Input Fields (QsnReadInp) operation or as a null field on a Read Modified Fields (QsnReadMDT) operation.

Format of the Field Control Word

The following table explains the valid field control words (FCW) for use with the QsnSetFld API.

Table 7. Field Control Words

FCW Value	Mnemonic	Description
X'80nn'	QSN_FCW_RESEQ	Entry field resequencing (used in the Read Input Fields (QsnReadInp) and Read Modified Fields (QsnReadMDT) APIs, and so forth). The *nn* specifies the next entry field in the sequence (*nn* can be X'00' to X'80'). See Resequencing.
X'8101'	QSN_FCW_MSR	Magnetic stripe reader entry field.
X'8102'	QSN_FCW_SLP	Selector light pen entry field.
X'8103'	QSN_FCW_MSR_SLP	Magnetic stripe reader and selector light pen entry field.
X'8106'	QSN_FCW_SLP_SA	Selector light pen and selectable attention entry field.
X'8200'	QSN_FCW_DBCS_ONLY	DBCS only entry field.
X'8220'	QSN_FCW_DBCS_PURE	DBCS Graphic DBCS entry field.
X'8240'	QSN_FCW_DBCS_EITHER	DBCS Either entry field.
X'8280' or X'82C0'	QSN_FCW_DBCS_OPEN or QSN_FCW_DBCS_OPEN_C0	DBCS open entry field.
X'84??'	QSN_FCW_TRANSPARENT	Transparency entry field (used in the Read Input Fields (QsnReadInp) and Read Modified Fields (QsnReadMDT) APIs, and so forth). The *??* indicates that these values are ignored.
X'8501'	QSN_FCW_FET	Forward edge trigger entry field. This provides the same function as Auto Enter specified in the FFW, except a unique AID is returned to the host when the field is exited. The state on the Auto Enter flag in the FFW is ignored.
X'8601'	QSN_FCW_CONT_FIRST	Continued entry field first segment.
X'8602'	QSN_FCW_CONT_LAST	Continued entry field last segment.
X'8603'	QSN_FCW_CONT_MIDDLE	Continued entry field middle segment.
X'88nn'	QSN_FCW_CP	Cursor progression entry field.

FCW Value	Mnemonic	Description
X'89nn'	QSN_FCW_HL	Highlighted entry field.
X'8Ann'	QSN_FCW_PDS	Pointer device selection entry field.
X'B140'	QSN_FCW_MOD11	Self Check Modulus 11 entry field.
X'B1A0'	QSN_FCW_MOD10	Self Check Modulus 10 entry field.

Valid Field Control Word and Device Capability Combinations

The following table explains the field control words (FCW) available for certain capabilities of the device. If one of the FCW values listed is given to QsnSetFld and the corresponding device capability is not supported, a CPFA306 will result. QsnSetFld, however, does not detect all invalid FCW combinations. See the Field control words (FCW) parameter for more information.

See the "Query 5250 (QsnQry5250) API" on page 18 to determine the capablilities supported by the current device.

Table 8. Valid Field Control Word and Device Capability Combinations

FCW Values	Display capablility required for FCW to be valid
QSN_FCW_CONT_FIRST, QSN_FCW_CONT_MIDDLE, QSN_FCW_CONT_LAST, QSN_FCW_CP, QSN_FCW_HL, QSN_FCW_PDS	ENPTUI
QSN_FCW_DBCS_ONLY, QSN_FCW_DBCS_EITHER, QSN_FCW_DBCS_OPEN, QSN_FCW_DBCS_OPEN_C0	DBCS
QSN_FCW_DBCS_PURE	Pure DBCS
QSN_FCW_TRANSPARENT	Transparent Data

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA306 E	Command not supported by current device.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA30A E	Field length &1 not valid.
CPFA30B E	Invalid starting address for field.
CPFA30C E	Maximum allowable number of fields exceeded.
CPFA30D E	Invalid screen attribute.
CPFA30E E	Invalid field format word.
CPFA314 E	Memory allocation error.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA332 E	Incorrect field control word.
CPFA333 E	Parameter &1 not positive integer value.

Message ID	Error Message Text
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33D E	Invalid screen attribute.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Set Field with CCSID (QsnSetFldCC) API

Omissible Parameter Group:

1	Field ID
Input	Binary(4)
2	Field length
Input	Binary(4)
3	Display Positions
Input	Binary(4)
4	CCSID
Input	Binary(4)
5	Row
Input	Binary(4)
6	Column
Input	Binary(4)
7	Field format word (FFW)
Input	Char(2)
8	Field control words (FCW)
Input	Char(*)
9	Number of field control words
Input	Binary(4)
10	Monochrome attribute
Input	Char(1)
11	Color attribute
Input	Char(1)
12	Command buffer handle
Input	Binary(4)
13	Low-level environment handle
Input	Binary(4)
14	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Set Field with CCSID (QsnSetFldCC) API defines a CCSID-capable input field on the screen at the given row and column. The following occurs when this command is issued to the control unit as a direct operation or when the buffer containing the command is written out:

- Any outstanding AID requests are cleared.
- The keyboard is locked.
- If there is an entry in the format table whose starting address is equal to the address for this field, then that entry is modified. The FFW of the existing entry is replaced by the new FFW and the previous screen starting attribute is overlaid with the new screen starting attribute. The ending screen attribute is not rewritten. All FCWs and the length parameter are ignored. See the 5250 data stream documentation for details.
- If no entry can be found in the table for the field being defined, a new entry will be added to the end of the table. However, the address must be greater than the ending address of the field currently defined last in the format table or an error will occur. If the new entry is valid, it will contain the field's FFW, the optional FCWs, and the field's starting and ending address. An error will occur if an attempt is made to define too many fields on the screen (see the 5250 data stream documentation for details).

The display address after this operation will be the starting field address minus 1 if row and column are specified as valid positive integers or if this is the first field specified within the current WTD command. Otherwise, the display address will be one position past the ending screen attribute.

This command corresponds indirectly to the 5250 Write to Display (WTD) command with a Set Buffer Address order and a Start of Field order if the row and column parameters are specified. Special FCWs inserted into the data stream by this API to make the field CCSID-capable. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Restrictions

The same restrictions apply as for the Write Data with CCSID (QsnWrtDtaCC) API, with the exception that the trailing field attribute can be written past the end of the screen. (It will be suppressed by the control unit.)

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

» The CCSID value given must be supported by the device or emulator, otherwise a CPF3BDE will be signaled. «

Authorities and Locks

None

Omissible Parameter Group

Field ID

INPUT; BINARY(4)

The field ID to be associated with this field. The value specified can be any nonzero integer value. For APIs that accept a field ID parameter, this value can be subsequently used instead of the row and column parameter to specify a screen address. If the given ID is already defined, this operation will redefine that field ID with the values specified. To remove a field ID definition, use the QsnDltFldId API.

If a previously defined field ID is supplied and some or all of the parameters are omitted, the field is defined using the current field definition values for those omitted parameters.

If this field is omitted or specified with a value of zero, then no field ID is associated with this field description.

Field length
INPUT; BINARY(4)

The number of bytes that this field being defined contains (a single character might be composed of multiple bytes). If no field ID is specified, the length must be a positive integer value greater than 1 for signed numeric fields and greater than 0 for all other field types. The entire field must fit on the display. If a field ID is specified with a nonzero value, the length may be 0, in which case a field will not be defined on the screen; however, this will associate the field definition with the specified field ID.

Display positions
INPUT; BINARY(4)

The number of positions on the display allowed for this field being defined (a single character may require more than one position on the display).

If this parameter is not specified, the default is the number given by the field length parameter.

CCSID
INPUT; BINARY(4)

The CCSID of the data that can be entered into this field. If the CCSID given is not supported by the device, a CPF3BDE is signaled.

If this parameter is omitted (zero is passed in as the CCSID), and the field ID parameter was omitted or previously undefined, the job CCSID is used. If the job CCSID is 65535, the default job CCSID is used instead.

Row INPUT; BINARY(4)

The row at which to define the field. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula `base+offset=actual`. The `base` is the row location of the top window border (0 for full screen) if `offset` is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if `offset` is negative. The `offset` is the row parameter value specified, and `actual` is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

The starting field address will be the row and column locations given if both parameters are specified. Otherwise, it will be the current display address plus 1. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1, prior to writing the initial screen attribute and the field definition. The ending field address for this field is the starting field address plus the number of character positions for the field.

If a field ID is supplied along with a row and column, the row and column parameters will be stored as specified. These parameters will be used as relative or actual screen positions on a subsequent operation, depending upon the window mode setting for the environment supplied with that operation.

If a previously undefined field ID is supplied with this operation, the row and column parameters must be specified. Also, the row and column parameters must both be specified or omitted; one cannot be specified if the other is omitted. A CPFA307 error occurs if an incorrect cursor position is specified. On some devices, row and column can both be specified as 1, which will cause the field to be defined at row 1, column 1, with a screen attribute of normal (X'20'). If this is the case, then any initial screen attribute parameters specified are ignored. This is only supported by certain devices. Whether or not this is supported can be determined by the Query 5250 (QsnQry5250) API.

Column
INPUT; BINARY(4)

The column at which to write the data. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula `base+offset=actual`. The `base` is the column location of the left window border (0 for full screen) if `offset` is positive, or the column location of the right window border (screen width plus 1 for full screen) if `offset` is negative. The `offset` is the column parameter value specified, and `actual` is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Field format word (FFW)
INPUT; CHAR(2)

The field format word is a 2-byte value that controls the type of the field being defined. QsnSetFld Table 6 (page 140) shows the field types, and the corresponding bit to be set for each type. To omit this parameter, specify X'00' in both characters of the parameter. You must specify this parameter to define an input field, and it is required if a field control word is specified.

Field control words (FCW)
INPUT; CHAR(*)

An array of 2-byte field control words. The field control words are 2-byte values that request certain functions to be performed. Table 10 (page 151) shows the valid field control word values, their function, and mnemonics for those values.

Note: The 5250 CCSID-based I/O specific FCWs are not allowed, because they are implicitly added as part of the QsnSetFldCC operation. If the CCSID-based I/O CCSID or Maximum data length FCWs are given here, a CPFA332 will result. FCWs will not be exhaustivly checked to see if they are formatted correctly or to see if the function requested is valid for the current device. However, some FCWs are checked against the support provided by the device and a CPFA306 signaled if an incompatibility is found. Table 11 (page 151) shows the display capablility and FCW combinations that are valid.

Errors not found here may be detected and reported when the FCW is required during subsequent command and keystroke processing. See the 5250 data stream documentation for further details about the meaning and use of these functions.

Number of field control words
INPUT; BINARY(4)

The number of control words in the field control word array. Omitting this parameter or specifying it with a value of 0 indicates that no field control words are specified with the FCW parameter. If this parameter is specified with a nonzero value, the FCW parameter is required; if the FCW parameter is omitted, a CPFA31E error is issued.

Monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. A screen attribute is required for defining a field on the screen; if this parameter is omitted and monochrome attributes are to be used, X'20' is assumed. The initial screen attribute is written one position to the left of the starting field address. The ending screen attribute (X'20') is supplied by the controller and written at the end-of-field address plus 1.

The monochrome attribute and color attribute parameters consist of 1 byte that will be used as the screen attribute for a monochrome or a color display, respectively. One of these parameters will be selected based on the underlying display type, and the other will be discarded. See Screen Attribute Characters for a description of the screen attribute values.

Color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. A screen attribute is required for defining a field on the screen; if this parameter is omitted and color attributes are to be used, X'20' is assumed. See Screen Attribute Characters for a description of the screen attribute values.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the starting field address will be the supplied location. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Return Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Field Control Word

The following table explains the valid field control words (FCW) for use with the QsnSetFldCC API.

Field Control Words		
FCW Value	**Mnemonic**	**Description**
X'80nn'	QSN_FCW_RESEQ	Entry field resequencing (used in the Read Input Fields (QsnReadInp) and Read Modified Fields (QsnReadMDT) APIs, and so forth). The *nn* specifies the next entry field in the sequence (*nn* can be X'00' to X'80'). See Resequencing.
X'8101'	QSN_FCW_MSR	Magnetic stripe reader entry field.
X'8102'	QSN_FCW_SLP	Selector light pen entry field.
X'8103'	QSN_FCW_MSR_SLP	Magnetic stripe reader and selector light pen entry field.
X'8106'	QSN_FCW_SLP_SA	Selector light pen and selectable attention entry field.
X'8501'	QSN_FCW_FET	Forward edge trigger entry field. This provides the same function as Auto Enter specified in the FFW, except a unique AID is returned to the host when the field is exited. The state on the Auto Enter flag in the FFW is ignored.
X'8601'	QSN_FCW_CONT_FIRST	Continued entry field first segment.
X'8602'	QSN_FCW_CONT_LAST	Continued entry field last segment.
X'8603'	QSN_FCW_CONT_MIDDLE	Continued entry field middle segment.
X'88nn'	QSN_FCW_CP	Cursor progression entry field.
X'89nn'	QSN_FCW_HL	Highlighted entry field.
X'8Ann'	QSN_FCW_PDS	Pointer device selection entry field.
X'B140'	QSN_FCW_MOD11	Self Check Modulus 11 entry field.

Field Control Words		
FCW Value	**Mnemonic**	**Description**
X'B1A0'	QSN_FCW_MOD10	Self Check Modulus 10 entry field.

Valid Field Control Word and Device Capability Combinations

The following table explains the field control words (FCW) available for certain capabilities of the device. If one of the FCW values listed is given to QsnSetFldCC and the corresponding device capability is not supported, a CPFA306 will result. QsnSetFldCC, however, does not detect all invalid FCW combinations. See the Field control words (FCW) parameter for more information.

See the "Query 5250 (QsnQry5250) API" on page 18 to determine the capablilities supported by the current device.

Valid Field Control Word and Device Capability Combinations	
FCW Values	**Display capablility required for FCW to be valid**
QSN_FCW_CONT_FIRST, QSN_FCW_CONT_MIDDLE, QSN_FCW_CONT_LAST, QSN_FCW_CP, QSN_FCW_HL, QSN_FCW_PDS	ENPTUI

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3BDE E	CCSID &1 not supported by API.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA306 E	Command not supported by current device.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA30A E	Field length &1 not valid.
CPFA30B E	Invalid starting address for field.
CPFA30C E	Maximum allowable number of fields exceeded.
CPFA30D E	Invalid screen attribute.
CPFA30E E	Invalid field format word.
CPFA314 E	Memory allocation error.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA332 E	Incorrect field control word.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33D E	Invalid screen attribute.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API Introduced: V5R2

Set Output Address (QsnSetOutAdr) API

Omissible Parameter Group:	
1	Field ID
Input	Binary(4)
2	Row
Input	Binary(4)
3	Column
Input	Binary(4)
4	Command buffer handle
Input	Binary(4)
5	Low-level environment handle
Input	Binary(4)
6	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Set Output Address (QsnSetOutAdr) API sets the current display address. Subsequent output operations that do not reset the display address use this address. If multiple Set Output Address (QsnSetOutAdr) operations are applied to the same command buffer, only the last QsnSetOutAdr operation is in effect.

This command corresponds indirectly to the 5250 Write to Display command (WTD) with a Set Buffer Address order. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Authorities and Locks

None

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. Either the field ID or the row and column parameters must be specified.

Row INPUT; BINARY(4)

The row at which to set the display address. This parameter is required if the field ID is not specified.

Column
INPUT; BINARY(4)

The column at which to set the display address. This parameter is required if the field ID is not specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the display address is set to the specified location. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA343 E	Output operation not done.

Message ID	Error Message Text
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Data (QsnWrtDta) API

Required Parameter Group:

1	Data
Input	Char(*)
2	Data length
Input	Binary(4)

Omissible Parameter Group:

3	Field ID
Input	Binary(4)
4	Row
Input	Binary(4)
5	Column
Input	Binary(4)
6	Starting monochrome attribute
Input	Char(1)
7	Ending monochrome attribute
Input	Char(1)
8	Starting color attribute
Input	Char(1)
9	Ending color attribute
Input	Char(1)
10	Command buffer handle
Input	Binary(4)
11	Low-level environment handle
Input	Binary(4)
12	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write Data (QsnWrtDta) API writes data to the display at a given row and column using standard attributes. If a command buffer is specified that does not contain a previous or current WTD command, one is implicitly added to the buffer using the control characters QSN_CC1_NULL and

QSN_CC2_UNLOCKBD. The display address after this operation will be one position past the last data byte written to the screen (including the ending screen attribute, if any).

This command corresponds indirectly to the 5250 Write to Display command (WTD) with a Set Buffer Address order if the row and column are specified. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Restrictions

If window mode is not set and the data (including attributes) exceeds the length of a row, the data will be wrapped to the next line. If bottom-to-top wrapping is not supported, a CPFA308 error occurs when the data (including attributes) exceeds the last row on the screen. If window mode is set, data that exceeds the width of the window will not be truncated or wrapped within the window, but will wrap across screen rows.

If the field ID given was created or last redefined by the QsnSetFldCC API, a CPFA346 will be signaled. This API should only be used to write data to fields that are not CCSID-capable. QsnWrtDta is only able to enforce this for fields that have an associated field ID, however.

Authorities and Locks

None

Required Parameter Group

Data INPUT; CHAR(*)

The data to be written to the screen. If the data being passed is graphic DBCS, the data must be enclosed by extended ideographic attributes. (Use the Write Data (QsnWrtDta) API to specify the data stream Write Extended Attribute order. See the 5250 Functions Reference, SA21-9247, for further details.)

Data length
INPUT; BINARY(4)

The number of characters contained in the data parameter.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. If neither the field ID nor the row and column parameters are specified, the current display address is used.

Row INPUT; BINARY(4)

The row at which to write the data. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the field ID and the row and column parameters are omitted, the data is written starting at the current display address. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1 prior to writing the data.

Row and column must both be specified or omitted; one cannot be specified if the other is omitted.

Column
INPUT; BINARY(4)

The column at which to write the data. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Starting monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. If this parameter is omitted and monochrome attributes are to be used, no initial attribute is written to the display for the data.

The monochrome and color attributes parameters are the initial and ending screen attributes: an initial and ending screen attribute to be used for a monochrome or a color display, respectively. One of these parameters will be selected based on the underlying display type, and the other will be discarded. Any of the attributes can be specified as a special value, X'00', indicating that no screen attribute should be written to the display. If the initial screen attribute is specified as an actual attribute, the data column, if specified, must be greater than or equal to 2. The initial screen attribute, if not X'00', will be written to the screen at the column previous to the first data character if row and column are specified, otherwise to the current display address. The ending screen attribute, if not X'00', will be written at the column directly after the last data character. See Screen Attribute Characters for a description of the screen attribute values.

Ending monochrome attribute
INPUT; CHAR(1)

The ending screen attribute for monochrome displays. If this parameter is omitted, and monochrome attributes are to be used, no ending attribute is written to the display for the data.

Starting color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no initial attribute is written to the display for the data.

Ending color attribute
INPUT; CHAR(1)

The ending screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no ending attribute is written to the display for the data.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the data is written to the screen at the specified location. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA30D E	Invalid screen attribute.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA33F E	Error occurred during data conversion.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA346 E	Operation for field ID &1 not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Data with CCSID (QsnWrtDtaCC) API

Required Parameter Group:

1	Data
Input	Char(*)
2	Data length
Input	Binary(4)

Omissible Parameter Group:

3	Field ID
Input	Binary(4)
4	CCSID
Input	Binary(4)
5	Row
Input	Binary(4)
6	Column
Input	Binary(4)
7	Starting monochrome attribute
Input	Char(1)
8	Ending monochrome attribute
Input	Char(1)
9	Starting color attribute
Input	Char(1)
10	Ending color attribute
Input	Char(1)
11	Command buffer handle
Input	Binary(4)
12	Low-level environment handle
Input	Binary(4)
13	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write Data with CCSID (QsnWrtDtaCC) API writes data to the display at a given row and column using standard attributes. If a command buffer is specified that does not contain a previous or current WTD command, one is implicitly added to the buffer using the control characters QSN_CC1_NULL and QSN_CC2_UNLOCKBD. The display address after this operation will be one position past the last data character position written to the screen (including the ending screen attribute, if any).

This command corresponds indirectly to the 5250 Write to Display command (WTD) with a Set Buffer Address order and Structured Field order if the row and column are specified. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Note: CDRA conversion is not performed upon this data.

Restrictions

If window mode is not set and the data (including attributes) exceeds the length of a row, the data will be wrapped to the next line. If bottom-to-top wrapping is not supported, a CPFA308 error occurs when the data (including attributes) exceeds the last row on the screen. If window mode is set, data that exceeds the width of the window will not be truncated or wrapped within the window, but will wrap across screen rows.

If the field ID given was created or last redefined by the QsnSetFld API, a CPFA346 will be signaled. This API should only be used to write data to CCSID-capable fields. QsnWrtDtaCC is only able to enforce this for fields that have an associated field ID, however.

QsnWrtDtaCC does not validate the data passed to it. If the data is invalid for some reason, an exception may occur if the control unit encounters improperly coded CCSID-based data. QsnWrtDta is not able to truncate data if it does not fit within the bounds of a field or low level environment window area as QsnWrtDta does for EBCDIC data.

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

» The CCSID value given must be supported by the device or emulator, otherwise a CPF3BDE will be signaled. «

Authorities and Locks

None.

Required Parameter Group

Data INPUT; CHAR(*)

The data in the CCSID given by the CCSID parameter to write to the screen.

Data length
INPUT; BINARY(4)

The number of bytes contained in the data parameter.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. If neither the field ID nor the row and column parameters are specified, the current display address is used. If this

parameter is specified with a nonzero value, the CCSID parameter is also ignored, and the data written to the screen is assumed have the same CCSID as the field ID.

CCSID
INPUT; BINARY(4)

The CCSID of the data to be written. If the CCSID given is not supported by the device, a CPF3BDE is signaled.

If this parameter is omitted (zero is passed in as the CCSID), and the field ID parameter was omitted or previously undefined, the job CCSID is used. If the job CCSID is 65535, the default job CCSID is used instead.

Row INPUT; BINARY(4)

The row at which to write the data. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula `base+offset=actual`. The `base` is the row location of the top window border (0 for full screen) if `offset` is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if `offset` is negative. The `offset` is the row parameter value specified, and `actual` is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the field ID and the row and column parameters are omitted, the data is written starting at the current display address. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1 prior to writing the data.

Row and column must both be specified or omitted; one cannot be specified if the other is omitted.

Column
INPUT; BINARY(4)

The column at which to write the data. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula `base+offset=actual`. The `base` is the column location of the left window border (0 for full screen) if `offset` is positive, or the column location of the right window border (screen width plus 1 for full screen) if `offset` is negative. The `offset` is the column parameter value specified, and `actual` is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Starting monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. If this parameter is omitted and monochrome attributes are to be used, no initial attribute is written to the display for the data.

The monochrome and color attributes parameters are the initial and ending screen attributes: an initial and ending screen attribute to be used for a monochrome or a color display, respectively. One of these parameters will be selected based on the underlying display type, and the other will be discarded. Any of the attributes can be specified as a special value, X'00', indicating that no screen attribute should be written to the display. If the initial screen attribute is specified as an actual attribute, the data column, if specified, must be greater than or equal to 2. The initial screen attribute, if not X'00', will be written to the screen at the column previous to the first data character if row and column are specified, otherwise to the current display address. The ending screen attribute, if not X'00', will be written at the column directly after the last data character. See Screen Attribute Characters for a description of the screen attribute values.

Ending monochrome attribute
INPUT; CHAR(1)

The ending screen attribute for monochrome displays. If this parameter is omitted, and monochrome attributes are to be used, no ending attribute is written to the display for the data.

Starting color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no initial attribute is written to the display for the data.

Ending color attribute
INPUT; CHAR(1)

The ending screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no ending attribute is written to the display for the data.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the data is written to the screen at the specified location. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3BDE E	CCSID &1 not supported by API.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA30D E	Invalid screen attribute.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.

Message ID	Error Message Text
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA33F E	Error occurred during data conversion.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA346 E	Operation for field ID &1 not valid.

Introduced: V5R2

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Structured Field Major (QsnWrtSFMaj) API

Required Parameter Group:

1 Major structure

Input Char(*)

2 Major structure length

Input Binary(4)

Omissible Parameter Group:

3 Field ID

Input Binary(4)

4 Row

Input Binary(4)

5 Column

Input Binary(4)

6 Command buffer handle

Input Binary(4)

7 Low-level environment handle

Input Binary(4)

8 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write Structured Field Major (QsnWrtSFMaj) API writes the major structure of a structured field, setting the display address to the given row and column. If a command buffer is specified that does not contain a previous or current WTD operation, one is implicitly added to the buffer using the control characters QSN_CC1_NULL and QSN_CC2_UNLOCKBD.

Use this API in conjunction with the Write Structured Field Minor (QsnWrtSFMin) API to construct a structured field operation. For indirect operations, the length contained in the minor structure data parameter is added to the stored length for this major structure for every indirect QsnWrtSFMin operation encountered directly after this operation. In this way, you need only calculate the length of each individual structure for constructing a structured field operation. See the 5494 Remote Control Unit Functions Reference, SC30-3533, for more information regarding structured fields.

This command corresponds indirectly to the 5250 Write to Display (WTD) command with a Set Buffer Address and a Write to Display Structured Field order if the row and column are specified. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in the buffer.)

Authorities and Locks

None

Restrictions

If window mode is not set and the data (including attributes) exceeds the length of a row, the data will be wrapped to the next line. If bottom-to-top wrapping is not supported, a CPFA308 error occurs when the data (including attributes) exceeds the last row on the screen.

Not all structured field types are supported by all devices. A negative response code is issued if an attempt is made to write a type to a device that does not support it.

Required Parameter Group

Major structure
INPUT; CHAR(*)

The major structure to be written to the screen. The data must consist of the entire major structure as documented in the 5494 Remote Control Unit Functions Reference, SC30-3533.

Major structure length
INPUT; BINARY(4)

The length of the major structure parameter. This is the length only and does not include any associated minor structure lengths.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. If neither the field ID nor the row and column parameters are specified, the current display address is used.

Row INPUT; BINARY(4)

The row at which to write the structure. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row

location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the field ID and the row and column parameters are omitted, the data is written starting at the current display address. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1 prior to writing the data.

Row and column must both be specified or omitted; one cannot be specified if the other is omitted.

Column
INPUT; BINARY(4)

The column at which to write the data. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the data is written to the screen at the specified location. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.

Message ID	Error Message Text
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA341 E	Length &2 of structure incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Structured Field Minor (QsnWrtSFMin) API

Required Parameter Group:

1 Minor structure

Input Char(*)

2 Minor structure length

Input Binary(4)

3 Command buffer handle

Input Binary(4)

Omissible Parameter Group:

4 Low-level environment handle

Input Binary(4)

5 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write Structured Field Minor (QsnWrtSFMin) API writes the minor structure of a structured field, incrementing the length field in the corresponding major structure by the length of this minor structure. The QsnWrtSFMin API must directly follow a QsnWrtSFMaj or QsnWrtSFMin operation to the given command buffer.

Use this API in conjunction with the Write Structured Field Major (QsnWrtSFMaj) API to construct a structured field operation. For indirect operations, the length contained in the minor structure data parameter is added to the stored length for this major structure for every indirect QsnWrtSFMin operation encountered directly after this operation. In this way, you need only calculate the length of each individual structure for constructing a structured field operation. See the *5494 Remote Control Unit Functions Reference*, SC30-3533, for more information regarding structured fields.

Authorities and Locks

None.

Restrictions

Not all structured field types are supported by all devices. A negative response code is issued if an attempt is made to write a type to a device that does not support it.

Required Parameter Group

Minor structure
INPUT; CHAR(*)

The minor structure to be written to the screen. The data must consist of the entire minor structure as documented in the 5494 Remote Control Unit Functions Reference, SC30-3533.

Minor structure length
INPUT; BINARY(4)

The length of the minor structure.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. The last operation stored in this command buffer must have been either a QsnWrtSFMaj or QsnWrtSFMin operation.

Omissible Parameter Group

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA308 E	Attempt to write data past end of display.
CPFA30D E	Invalid screen attribute.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33B E	Incorrect operation before QsnWrtSFMin.
CPFA33C E	Undefined field ID &1.
CPFA341 E	Length &2 of structure incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write to Display (QsnWTD) API

Required Parameter Group:

1	Control character byte 1
Input	Char(1)
2	Control character byte 2
Input	Char(1)

Omissible Parameter Group:

3	Command buffer handle
Input	Binary(4)
4	Low-level environment handle
Input	Binary(4)
5	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write to Display (QsnWTD) API issues a Write to Display (WTD) command. A WTD command is used to indicate the beginning of a series of output operations. For most of the DSM screen output operations, the control unit requires that a WTD be issued prior to the actual operation itself, in the same I/O operation. Multiple output operations can be sent with a given WTD command by using a command buffer. For direct operations, such output APIs, implicitly issue a WTD command. For indirect operations, a WTD is added to the buffer if one does not exist already. Each time a WTD command is received, the control unit implicitly sets the display address to row 1, column 1, prior to issuing the operation.

For example, if two direct QsnWrtDta operations are performed in succession and the second operation does not specify the row and column parameters, the data is written to row 1, column 1, and not the display address following the first QsnWrtDta operation. If two indirect QsnWrtDta operations are issued to an empty command buffer, the first QsnWrtDta operation causes a WTD command to be added to the buffer, but the second QsnWrtDta operation can use the existing command. In this case, if no row and column are specified, the display address used for the second operation will be the one after the first operation is issued. If an intervening operation that does not require a WTD, such as a QsnClrScr or a QsnSetErr, is added to the command buffer between two operations that do require a WTD command, a second WTD command is added to the buffer for the second operation.

The screen output APIs that require a WTD command and that correspond to the WTD command **orders** (as described in 5250 Functions Reference, SA21-9247) are as follows:

Set Output Address (QsnSetOutAdr)

Write Data (QsnWrtDta)

Write Transparent Data (QsnWrtTDta)

Pad for N Positions (QsnWrtPad)

Pad between Two Screen Addresses (QsnWrtPadAdr)

Set Field (QsnSetFld)

Set Cursor Address (QsnSetCsrAdr)

Insert Cursor (QsnInsCsr)

The cursor position is not affected if the keyboard is unlocked when command processing begins and is not locked during command processing, or if a parameter error is detected. If specified by control character byte 1, the cursor will be moved to the insert cursor or default location when the keyboard is unlocked.

This operation corresponds directly to the 5250 Write to Display command. If this is an indirect operation, this operation will issue a new WTD command to the command buffer.

Authorities and Locks

None

Required Parameter Group

Control character byte 1
INPUT; CHAR(1)

The operation for the display to perform prior to processing the Write to Display command. See Control Characters for a description of the control character values.

Control character byte 2
INPUT; CHAR(1)

The operation for the display to perform after the Write to Display command has been processed. See Control Characters for a description of the control character values.

Omissible Parameter Group

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is omitted or specified as 0, this is a direct operation and the command is sent to the display. Otherwise, this is an indirect operation and the command is stored in the command buffer without an I/O operation taking place.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA303 E	Error occurred for screen I/O operation.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA31C E	Incorrect value for control character byte &1.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA334 E	Low level environment handle incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Transparent Data (QsnWrtTDta) API

Required Parameter Group:

1	Data
Input	Char(*)
2	Data length
Input	Binary(4)

Omissible Parameter Group:

3	Field ID
Input	Binary(4)
4	Row
Input	Binary(4)
5	Column
Input	Binary(4)
6	Starting monochrome attribute
Input	Char(1)
7	Ending monochrome attribute
Input	Char(1)
8	Starting color attribute
Input	Char(1)
9	Ending color attribute
Input	Char(1)
10	Command buffer handle
Input	Binary(4)
11	Low-level environment handle
Input	Binary(4)
12	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write Transparent Data (QsnWrtTDta) API writes *n* bytes of transparent data to the display.

This command allows transmission of data with any value (X'00' to X'FF') to the display screen. If the data destination is a 5250 display, and if the data X'04', X'11', or X'FF' is transmitted, unpredictable

results occur. Note that if DBCS characters are included in the data, the host system does not perform IGC extension character processing. However, SI/SO characters will be processed correctly.

The display address after this operation is one position past the last data byte written to the screen.

This command corresponds indirectly to the 5250 Write to Display (WTD) command with a Set Buffer Address and a Transparent Data order. (For an indirect operation, a WTD is placed in the command buffer only if one does not already exist in that buffer.)

Authorities and Locks

None

Restrictions

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it. Additional restrictions apply as for the Write Data (QsnWrtDta) API.

If the field ID given was created or last redefined by the QsnSetFldCC API, a CPFA346 will be signaled. This API should only be used to write data to fields that are not CCSID-capable. QsnWrtTDta is only able to enforce this for fields that have an associated field ID, however.

Required Parameter Group

Data INPUT; CHAR(*)

The data to be written to the screen.

Data length
INPUT; BINARY(4)

The number of bytes of data to be written.

Omissible Parameter Group

Field ID
INPUT; BINARY(4)

The field ID indicating the field at which to set the display address. If this parameter is specified with a nonzero value, the row and column parameters are ignored and the row and column values corresponding to the field ID are used to set the display address. If neither the field ID nor the row and column parameters are specified, the current display address is used.

Row INPUT; BINARY(4)

The row at which to write the data. The row parameter must refer to a row no greater than the current screen or window mode height (if window mode is enabled). The actual screen row used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the row location of the top window border (0 for full screen) if offset is positive, or the row location of the bottom window border (screen height plus 1 for full screen) if offset is negative. The offset is the row parameter value specified, and actual is the actual screen row to be used. A CPFA307 error occurs if an incorrect row value is specified.

If both the field ID and the row and column parameters are omitted, the data is written starting at the current display address. If this is the case and the command is a direct operation, or the buffer specified does not contain a preceding output operation that sets the display address, the current display address is set to row 1, column 1, prior to writing the data. Row and column must both be specified or omitted; one cannot be specified if the other is omitted.

Column
INPUT; BINARY(4)

The column at which to write the data. The column parameter must refer to a column no greater than the current screen or window mode width (if window mode is on). The actual screen column used for a screen I/O operation is calculated using the formula base+offset=actual. The base is the column location of the left window border (0 for full screen) if offset is positive, or the column location of the center window border (screen width plus 1 for full screen) if offset is negative. The offset is the column parameter value specified, and actual is the actual screen column to be used. A CPFA307 error occurs if an incorrect column value is specified.

Starting monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. If this parameter is omitted and monochrome attributes are to be used, no initial attribute is written to the display for the data. See Screen Attribute Characters for a description of the screen attribute values. The attribute parameters are specified with the same effect as for the QsnWrtDta operation.

Ending monochrome attribute
INPUT; CHAR(1)

The ending screen attribute for monochrome displays. If this parameter is omitted and monochrome attributes are to be used, no ending attribute is written to the display for the data.

Starting color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no initial attribute is written to the display for the data.

Ending color attribute
INPUT; CHAR(1)

The ending screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no ending attribute is written to the display for the data.

Command buffer handle
INPUT; BINARY(4)

A handle for the command buffer in which to store the command. If this parameter is specified, this is an indirect operation; the command is stored in the command buffer without an I/O operation taking place. If this parameter is omitted or specified with a zero value, this is a direct operation; the data is written to the screen at the specified location.

Low-level environment handle
INPUT; BINARY(4)

The low-level environment that the operation applies to. If this parameter is omitted or given with a value of zero, the default low-level environment is used.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA301 E	Command buffer is full.
CPFA304 E	Data-stream error &1 reported for screen I/O operation.
CPFA305 E	Cannot add operation to command buffer.
CPFA306 E	Command not supported by current device.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA31E E	Required parameter &1 omitted.
CPFA31D E	Attempt to write outside of window area.
CPFA31E E	Required parameter &1 omitted.
CPFA331 E	Buffer handle incorrect.
CPFA333 E	Parameter &1 not positive integer value.
CPFA334 E	Low level environment handle incorrect.
CPFA335 E	Screen address parameter error.
CPFA33C E	Undefined field ID &1.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA346 E	Operation for field ID &1 not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Window Services APIs

The window services APIs consist of two functional groups: the window builder and window manager services. The window builder APIs provide the services needed to create, delete, move, and resize windows. The window builder services include the window manipulation and query APIs, and the window I/O APIs. The window manager APIs provide the services needed to manage multiple windows, support I/O to several active windows, and switch between windows.

Window Services APIs include:

- "Window Manipulation and Query APIs"
- "Window I/O APIs" on page 203
- "Window Manager Services APIs" on page 212

See "Using Window Services APIs" on page 297 for additional information.

"Dynamic Screen Manager APIs," on page 1 | APIs by category

Window Manipulation and Query APIs

The window manipulation and query APIs are:

- "Change Window (QsnChgWin) API" on page 176 (QsnChgWin) changes the description for a window.
- "Create a Window (QsnCrtWin) API" on page 178 (QsnCrtWin) creates a window.
- "Initialize Window Description (QsnInzWinD) API" on page 189 (QsnInzWinD) initializes a window description with default values.

- "Move Window (QsnMovWin) API" on page 191 (QsnMovWin) moves a window to a new screen location.
- "Move Window by User (QsnMovWinUsr) API" on page 193 (QsnMovWinUsr) moves a window to a new screen location specified by the user.
- "Resize Window (QsnRszWin) API" on page 194 (QsnRszWin) changes the size of a window.
- "Resize Window by User (QsnRszWinUsr) API" on page 196 (QsnRszWinUsr) changes the size of a window according to user-specified cursor movements.
- "Retrieve Window Data (QsnRtvWinDta) API" on page 198 (QsnRtvWinDta) returns a pointer to the user data for the given window.
- "Retrieve Window Description (QsnRtvWinD) API" on page 199 (QsnRtvWinD) retrieves a copy of the description for a window.
- "Set Window Services Attributes (QsnSetWinAtr) API" on page 201 (QsnSetWinAtr) sets the default attributes for the window services.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Change Window (QsnChgWin) API

Required Parameter Group:

1	Window handle	
Input	Binary(4)	
2	Window description	
Input	Char(*)	
3	Length of window description	
Input	Binary(4)	

Omissible Parameter:

4	Error code	
I/O	Char(*)	

Returned Value:

	Return code	
Output	Binary(4)	

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Change Window (QsnChgWin) API changes the window description for the given window. The size cannot be changed for a window that contains DBCS data.

Authorities and Locks

Exit Routine Authority
*EXECUTE

Required Parameter Group

Window handle
INPUT; BINARY(4)

A handle for the window that will have its description changed.

Window description
INPUT; CHAR(*)

The new window description for the given window. The format of the window description is shown in "Format of the Window Description" on page 180.

Length of window description
Input; BINARY(4)

The length of the window description parameter.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA314 E	Memory allocation error.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A1 E	Window description value is incorrect.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Create a Window (QsnCrtWin) API

Required Parameter Group:

1 Window description

Input Char(*)

2 Length of window description

Input Binary(4)

Omissible Parameter Group:

3 User extension information

Input Char(*)

4 Length of user extension information

Input Binary(4)

5 Start window

Input Char(1)

6 Low-level environment description

Input Char(*)

7 Length of low-level environment description

Input Binary(4)

8 Window handle

Output Binary(4)

9 Error code

I/O Char(*)

Returned Value:

Window handle

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Create a Window (QsnCrtWin) API creates a window and returns a handle for that window. The window must be deleted using the Delete Low-Level Environment (QsnDltEnv) API. Whenever a window is made current, the window mode is set to the usable area of the window.

Authorities and Locks

Exit Routine Authority
*EXECUTE

Required Parameter Group

Window description
INPUT; CHAR(*)

The window description defines the attributes for the window. The format of the window description is shown in "Format of the Window Description" on page 180.

Length of window description
INPUT; BINARY(4)

The length of the window description parameter.

Omissible Parameter Group

User extension information
INPUT; CHAR(*)

The user extension information is used to associate data and exit routines with the window. This parameter is required if the length of user extension information parameter is supplied. This essentially enables the object-oriented programming concept of inheritance, allowing the window to be extended in a natural way. The user extension information cannot be changed once the window has been created. The format of this parameter is shown in "Format of the Window User Extension Information" on page 185.

Length of user extension information
INPUT; BINARY(4)

The length of the user extension information parameter.

Start window
Input; CHAR(1)

Whether the window should be displayed on the screen when it is allocated. The possible values are:

0	The window is not displayed on the screen when it is allocated. You must use the Start a Window (QsnStrWin) API to start the window.
1	The window is displayed on the screen when it is allocated. This is the default if this parameter is omitted.

Low-level environment description
INPUT; CHAR(*)

The low-level environment description defines the operating environment for low-level operations used to create and manipulate the window. This parameter is required if the length of the low-level environment description parameter is supplied. The format of the low-level environment description is shown in "Format of the Low-Level Environment Description" on page 10. If this parameter is omitted, a low-level environment will be created with default values.

Length of low-level environment description
INPUT; BINARY(4)

The length of the low-level environment description parameter.

Window handle
OUTPUT; BINARY(4)

The variable containing the handle for the window created after the QsnCrtWin API has completed. This handle can be used across activation groups if the activation group in which the handle was created is still active.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Window handle

OUTPUT; BINARY(4)

This API returns the value for the window handle parameter or a -1 if an error occurred during processing.

Restrictions

Windows where the associated low-level environment indicates DBCS support cannot be resized. GUI windows must fit within the screen boundary. » This includes the leading left border attribute and the right border continuation attribute. GUI windows must have a border and the associated left and right border attributes for the left and right borders. « The concept of current and noncurrent window border attributes does not apply to GUI windows. No error-checking is performed for GUI-specific fields. The fields are passed to the control unit, as specified, and any errors will be detected by the control unit.

Format of the Window Description

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Row location of upper left corner of window border
4	4	BINARY(4)	Column location of upper left corner of window border
8	8	BINARY(4)	Number of rows within the window
12	C	BINARY(4)	Number of columns within the window
16	10	BINARY(4)	Minimum number of rows within the window
20	14	BINARY(4)	Minimum number of columns within the window
24	18	BINARY(4)	Maximum number of rows within the window
28	1C	BINARY(4)	Maximum number of columns within the window
32	20	CHAR(1)	Full-screen flag for the window
33	21	CHAR(3)	Window display attributes for a monochrome display
36	24	CHAR(3)	Window display attributes for a color display
39	27	CHAR(1)	Border flag for the window
40	28	CHAR(1)	Border attributes flag for the window
41	29	CHAR(1)	Leading attribute flag for the window
42	2A	CHAR(1)	» Right continuation attribute flag for the window «
43	2B	CHAR(1)	Message line flag for the window
44	2C	CHAR(1)	Upper left border character
45	2D	CHAR(1)	Top border character
46	2E	CHAR(1)	» Upper right border character «
47	2F	CHAR(1)	Left border character
48	30	CHAR(1)	» Right border character «
49	31	CHAR(1)	Lower left border character
50	32	CHAR(1)	Bottom border character
51	33	CHAR(1)	» Lower right border character «

Offset		Type	Field
Dec	Hex		
52	34	CHAR(1)	GUI support flag for the window
53	35	CHAR(1)	Flag byte 1 (GUI only)
54	36	CHAR(1)	Flag byte 2 (GUI only)
55	37	CHAR(1)	Reserved (GUI only). The default is X'00'.
56	38	CHAR(1)	Border flags (GUI only)
57	39	CHAR(1)	Window title flags (GUI only)
58	3A	CHAR(1)	Monochrome title attribute
59	3B	CHAR(1)	Color title attribute
60	3C	CHAR(1)	Reserved (GUI only). The default is X'00'.
61	3D	CHAR(3)	Reserved. This field must be set to X'00'.
64	40	BINARY(4)	Offset to title text
68	44	BINARY(4)	Length of title text
72	48	BINARY(4)	Reserved. This field must be set to X'00'.
*	*	CHAR(*)	Title text

Field Descriptions

In the following descriptions, the default value refers to the value set by the Initialize Window Description (QsnInzWinD) API.

The GUI-only fields in the following descriptions refer to fields within the data stream for the Create Window command major and minor structures. See the 5494 Remote Control Unit Functions Reference, SC30-3533, manual for details.

Border attributes flag for the window. » Whether or not the window border has left and right border attributes. « (See DSM Window Layout (page 298) in Window Services APIs.) The possible values are:

0	Window border has no attributes.
1	Window border has attributes. This is the default.

For GUI windows, 1 must be specified.

Border flag for the window. This flag indicates whether or not the window has a border. (See DSM Window Layout (page 298) in Window Services APIs.) The possible values are:

0	Window has no border.
1	Window has a border. This is the default.

If this field is set to 0, the border attributes are not written to the screen, regardless of the values of the other fields.

For GUI windows, 1 must be specified.

Border flags (GUI only). Determines if border presentation characters are used (byte 3 of the border presentation minor structure of the GUI Create Window command). The default is X'80'.

Bottom border character. The character used for the bottom border. The default is X'00' for all border characters. The default border character used for non-GUI windows is '.'. For GUI windows, the default GUI character is used.

Color title attribute. The display attribute to precede the title on a color display. The default is X'20'. If this attribute is not valid, it is ignored.

Column location of upper left corner of window border. The column number of the upper left corner of the window. If the window has a leading window attribute, the first window column is two greater than the value specified in this field; otherwise, it is one greater. This must be a positive integer value greater than or equal to 0. For GUI windows, the minimum value must be 2. Otherwise, it must be a positive integer value greater than or equal to 0. For non-GUI windows, if 0 or 1 is specified, the left border of the window or the leading border attribute, respectively, will not be displayed on the screen unless the window is moved. The default value is such that the maximum-sized window will be displayed, with border and attributes, given the other window description attributes (for example, window border attributes). If the window is a full-screen window, this field is ignored.

Flag byte 1 (GUI only). Byte 5 of the GUI Create Window command major structure. The default is X'00'.

Flag byte 2 (GUI only). Byte 6 of the GUI Create Window command major structure. The default is X'00'.

Full-screen flag for the window. Indicates whether or not this is a full-screen window. (A full-screen window cannot be moved or resized.) The possible values are:

0	Window is not a full-screen window. This is the default.
1	Window is a full-screen window.

GUI support flag for the window. This flag indicates whether or not GUI support should be used to build the window if the underlying device supports it. GUI windows always include a leading and trailing border attribute. The possible values are:

0	Do not use GUI support.
1	Use GUI support if the underlying device supports it. This is the default.

Leading attribute flag for the window. This flag indicates whether or not the window has a leading attribute. (See DSM Window Layout (page 298) in Window Services APIs.) The possible values are:

0	Window has no leading attribute byte.
1	Window has a leading attribute byte. This is the default.

For GUI windows, 1 must be specified.

Left border character. The character used for the left border. The default is X'00' for all border characters. The default border character used for non-GUI windows is ':'. For GUI windows, the default GUI character is used.

Length of title text. The length of the title text. The default value is 0.

Lower left border character. The character used for the lower left border. The default is X'00' for all border characters. The default border character used for non-GUI windows is ':'. For GUI windows, the default GUI character is used.

» **Lower right border character.** The character used for the lower right border. « The default is X'00' for all border characters. The default border character used for non-GUI windows is ':'. For GUI windows, the default GUI character is used.

Maximum number of columns within the window. A value of 0, which is the default, indicates this value is the same as the maximum number of columns for the device in its current mode. If the window is a full-screen window, this field is ignored.

Maximum number of rows within the window. A value of 0, which is the default, indicates this value is the same as the maximum number of rows for the device in its current mode. If the window is a full-screen window, this field is ignored.

Message line flag for the window. This flag indicates whether or not the window has a message line. (See DSM Window Layout (page 298) in Window Services APIs). The possible values are:

0	Window does not have a message line
1	Window has a message line. This is the default.

Minimum number of columns within the window. The minimum value allowed is 1. This is the default. If the window is a full-screen window, this field is ignored.

Minimum number of rows within the window. The minimum value allowed is 1. This is the default. If the window is a full-screen window, this field is ignored.

Monochrome title attribute. The display attribute to precede the title on a monochrome display. The default is X'20'. If this attribute is not valid, it is ignored.

Number of columns within the window. The number of columns in the window, from the first window column to the last. This excludes the left and right border, and the border and window attributes. (See DSM Window Layout (page 298) in Window Services APIs.) A value of 0, which is the default, indicates this value is the maximum number of columns that can be defined given the other window description attributes (for example, window border attributes). The minimum allowed number of columns is 1. If the window is a full-screen window, this field is ignored.

Number of rows within the window. The number of rows in the window, from the first window row to the last. This includes the message line, if specified. (See DSM Window Layout (page 298) in Window Services APIs.) A value of 0, which is the default, indicates this value is the maximum number of rows that can be defined given the other window description attributes (for example, window message line). The minimum allowed number of rows is 1. If the window is a full-screen window, this field is ignored.

Offset to title text. The offset for the window title text. The default value is 0.

Right border character. The character used for the right border. The default is X'00' for all border characters. The default border character used for non-GUI windows is ':'. For GUI windows, the default GUI character is used.

Right continuation attribute flag for the window. Whether or not the window has a right continuation attribute (see DSM Window Layout (page 298) in Window Services APIs). The possible values are:

0	Window has no right continuation attribute byte.
1	Window has a right continuation attribute byte. This is the default. For GUI windows, 1 must be specified.

The right continuation attribute used is X'20', which is green for color displays and normal attribute for monochrome displays.

Row location of upper left corner of window border. The row number of the upper left corner of the window. The first window row is one greater than the value specified in this field. This must be a

positive integer value greater than or equal to 0. For GUI windows, the minimum value must be 1. Otherwise, it must be a positive integer value greater than or equal to 0. For non-GUI windows, if 0 is specified, the top border of the window will not be displayed on the screen unless the window is moved. The default value is such that the maximum-sized window will be displayed, with border and attributes, given the other window description attributes (for example, window border attributes). If the window is a full-screen window, this field is ignored.

Title text. The text for the window title, which is written in the top border of the window. If the title text is too long to fit in the window border, it is truncated. Otherwise, it is centered in the window border. You can add padding (extra blanks beside the text) to specify left or center justification for the title. If an attribute is specified for the title text, the window border attribute is placed after the title text. This field is ignored if the window does not have a top border.

The default is no title text.

Top border character. The character used for the top border. The default is X'00' for all border characters. The default border character used for non-GUI windows is '.'. For GUI windows, the default GUI character is used.

Upper left border character. The character used for the upper left border. The default is X'00' for all border characters. The default border character used for non-GUI windows is '.'. For GUI windows, the default GUI character is used.

Upper right border character. The character used for the upper right border. The default is X'00' for all border characters. The default border character used for non-GUI windows is '.'. For GUI windows, the default GUI character is used.

Window display attributes for a color display. The window display attributes for a color display. The first character is the attribute for the window border when the window is not current, the second for when the window is current, and the third for the leading window attribute. All bytes may contain the same value. The special value X'00' can be used to indicate that no screen attribute should be used for the given character. The first attribute is ignored for GUI windows, which only use the second attribute. If X'00' is specified as the second attribute for a GUI window, the default GUI border attribute will be used. Both the current and noncurrent border attributes must be either X'00' or a valid attribute. For example, it is incorrect to specify the current attribute field X'00' and the noncurrent attribute field with a valid attribute.

The default values for these fields are those specified by the window services mode description (see "Format of the Window Services Attribute Description" on page 202).

Window display attributes for a monochrome display. The window display attributes for a monochrome display. The first character is the attribute for the window border when the window is not current, the second for when the window is current, and the third is for the leading window attribute. All bytes may contain the same value. The special value X'00' can be used to indicate that no screen attribute should be used for the given character. The first attribute is ignored for GUI windows, which only use the second attribute. If X'00' is specified as the second attribute for a GUI window, the default GUI border attribute will be used. Both the current and noncurrent border attributes must be either X'00' or a valid attribute. For example, it is incorrect to specify the current attribute field X'00' and the noncurrent attribute field with a valid attribute.

The default values for these fields are those specified by the window services mode description (see "Format of the Window Services Attribute Description" on page 202).

Window title flags (GUI only). Byte 3 of the window title minor structure of the Create Window command. The default is X'00'.

Format of the Window User Extension Information

Offset		Type	Field
Dec	Hex		
0	0	PTR(SPP)	User data associated with window
16	10	PTR(PP)	Exit routine to call for Change Window (QsnChgWin) API
32	20	PTR(PP)	Exit routine to call for Delete Low-Level Environment (QsnDltEnv) API
48	30	PTR(PP)	Exit routine for QsnMovWin, Move Window by User (QsnMovWinUsr), Resize Window (QsnRszWin), or Resize Window by User (QsnRszWinUsr) APIs
64	40	PTR(PP)	Exit routine for Display Window (QsnDspWin) API
80	50	PTR(PP)	Exit routine for Set Current Window (QsnSetCurWin) API

Field Descriptions

Exit routine for Change Window (QsnChgWin) API. This exit routine is called after the window is changed. If the window is redrawn, it is called after the window is redrawn.

Exit routine for Delete Low-Level Environment (QsnDltEnv) API. The exit routine to call when a window is deleted using the Delete Low-Level Environment (QsnDltEnv) API.

Exit routine for Display Window (QsnDspWin) API. The exit routine to call immediately before the window is drawn or redrawn. The following APIs may cause the window to be redrawn: QsnCrtWin, QsnStrWin, QsnChgWin, QsnMovWin, QsnMovWinUsr, QsnRszWin, QsnRszWinUsr, QsnDspWin, and QsnSetCurWin.

Exit routine for move or resize window APIs. The exit routine to call when window coordinates are changed using the QsnMovWin, Move Window by User (QsnMovWinUsr), Resize Window (QsnRszWin), or Resize Window by User (QsnRszWinUsr) APIs. This exit routine is called after the window is redrawn.

Exit routine for Set Current Window (QsnSetCurWin) API. The exit routine to call whenever a window is made current by one of the following APIs: QsnCrtWin, QsnStrWin, or QsnSetCurWin. This exit routine is called after the window is drawn or redrawn.

User data associated with window. A pointer to any data that the user wants to associate with this window.

Window Exit Routines

Exit routines are user-supplied functions with a defined interface. The routines are called from certain APIs and allow the programmer to attach additional function to those APIs. For instance, if fields have been set up in a window, a Change Coordinates exit routine could be supplied to move the fields if the window is moved.

Exit Routine Error Handling

If an exception occurs during the processing of an exit routine, the exception is ignored and processing continues. A CPFA318 will be issued as a diagnostic message only. You can explicitly handle errors in an exit routine by adding an exception handler to the routine.

Change Window Exit Routine

This exit routine, if specified on the user extension information, is called when the window is changed. The following parameter is passed to the exit routine:

Parameter Passed to Exit Routine	
1	Window handle
Input	Binary(4)

Change Window Exit Routine Parameter

Window handle
INPUT; BINARY(4)

The window that was changed.

Delete Window Exit Routine

This exit routine, if specified on the user extension information, is called when the window is deleted. The following parameter is passed to the exit routine:

Parameter Passed to Exit Routine	
1	Window handle
Input	Binary(4)

Delete Window Exit Routine Parameter

Window handle
INPUT; BINARY(4)

The window that was deleted.

Change Window Coordinates Exit Routine

This exit routine, if specified on the user extension information, is called when the move or resize APIs are called, after the window has been successfully moved or resized, but before the window is drawn on the screen. For this reason, you should not use this exit routine to draw anything in the window. The draw exit routine is called when the window is moved or resized. The following parameters are passed to the exit routine:

Parameters Passed to Exit Routine	
1	Window handle
Input	Binary(4)
2	Top border offset
Input	Binary(4)
3	Left border offset
Input	Binary(4)
4	Bottom border offset
Input	Binary(4)
5	» Right border offset «
Input	Binary(4)

Change Window Coordinates Exit Routine Parameters

Window handle
INPUT; BINARY(4)

The window for which the coordinates were changed.

Top border offset
INPUT; BINARY(4)

The offset, in screen rows, from the previous top window border to the current top window border (after the window coordinates have been changed). It can be positive, negative, or zero, depending on how the top window border was changed. For example, if the top border was moved down two rows, this value would be 2; if it was moved up 4 rows, this value would be -4; if the top row was not changed, this value would be 0.

Left border offset
INPUT; BINARY(4)

The offset, in screen columns, from the previous left window border to the current left window border (after the window coordinates have been changed). It can be positive, negative, or zero, depending on how the left window border was changed. » For example, if the left border was moved two columns to the right, this value would be 2; « if it was moved 4 columns to the left, this value would be -4, and if the left column was not changed, this value would be 0.

Bottom border offset
INPUT; BINARY(4)

The offset, in screen rows, from the previous bottom window border to the current bottom window border (after the window coordinates have been changed). It can be positive, negative, or zero, depending on how the bottom window border was changed. For example, if the border was moved down two rows, this value would be 2; if it was moved up 4 rows, this value would be -4; if the bottom row was not changed, this value would be 0.

» **Right border offset**
INPUT; BINARY(4)

The offset, in screen columns, from the previous right window border to the current right window border (after the window coordinates have been changed). It can be positive, negative, or zero, depending on how the right window border was changed. For example, if the right border was moved two columns to the right, this value would be 2; if it was moved 4 columns to the left, this value would be -4; if the right column was not changed, this value would be 0. «

Draw Window Exit Routine

This exit routine, if specified on the user extension information, is called when the Display Window (QsnDspWin) API is called, before the window is drawn. Because the exit routine is called before the window is drawn, you should only write inside the window using the command buffer parameter. Direct operations should not be used for the exit routine. The following parameters are passed to the exit routine:

Parameters Passed to Exit Routine	
1	Window handle
Input	Binary(4)
2	Command buffer
Input	Binary(4)

Draw Window Exit Routine Parameters

Window handle
INPUT; BINARY(4)

The window to be drawn.

Command buffer
INPUT; BINARY(4)

The command buffer used to store the commands that re-create the window contents. The contents of the command buffer are written to the screen along with the window border. This allows the window and its contents to be redrawn in a single I/O operation.

Current Window Exit Routine

This exit routine, if specified on the user extension information, is called when the window is made current through the Set Current Window (QsnSetCurWin) API. The following parameter is passed to the exit routine:

Parameter Passed to Exit Routine	
1	Window handle
Input	Binary(4)

Current Window Exit Routine Parameter

Window handle
INPUT; BINARY(4)

A handle for the window that is made current.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.

Message ID	Error Message Text
CPFA314 E	Memory allocation error.
CPFA318 E	Error calling exit routine.
CPFA327 E	Low level environment description value incorrect.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A1 E	Window description value is incorrect.
CPFA3AB E	The value for &1 must be '0' or '1'.

Additional errors may be generated by this API. They are listed in "Error Messages" on page 15 under the Create Low-Level Environment (QsnCrtEnv) API.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Initialize Window Description (QsnInzWinD) API

Required Parameter Group:

1 Window description

Output Char(*)

2 Length of window description

Input Binary(4)

Omissible Parameter:

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

TheInitialize Window Description (QsnInzWinD) API initializes a window description with default values. Unless otherwise specified in the window description (see "Format of the Window Description" on page 180), pointer fields are set to the null pointer, numeric fields to 0, character flag fields to 0, and other character fields to blanks. For example, the default value for the border flag is 1, so this field will be set to 1.

Authorities and Locks

Exit Routine Authority
*EXECUTE

Required Parameter Group

Window description
OUTPUT; CHAR(*)

The window description to be initialized.

Length of window description
INPUT; BINARY(4)

The length of the window description parameter.

Omissible Parameter Group

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Move Window (QsnMovWin) API

Required Parameter Group:	
1	Window handle
Input	Binary(4)
2	Upper left row
Input	Binary(4)
3	Upper left column
Input	Binary(4)
Omissible Parameter:	
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The QsnMovWin API moves the window to the new upper left coordinate (upper left row, upper left column) specified. If the window can fit within the display at the location specified, it is moved to the new position. If a Change Window Coordinates exit routine is specified on the window description, it is called after the window is successfully moved. If the window is a full screen window, the API will complete successfully, but the window will not be moved.

Authorities and Locks

None

Required Parameter Group

Window handle
INPUT; BINARY(4)

A handle for the window to be moved.

Upper left row
INPUT; BINARY(4)

The absolute screen row for the new upper left corner of the window.

Upper left col
INPUT; BINARY(4)

The absolute screen column for the new upper left corner of the window.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A2 E	Window does not fit on screen.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Move Window by User (QsnMovWinUsr) API

Required Parameter:

1 Window handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Move Window by User (QsnMovWinUsr) API allows a window to be moved on the screen to a new location specified by the user. The API positions the cursor at the upper left corner of the window and prompts the user to move the cursor to the new position for the upper left corner. The prompt is displayed only if a message line has been defined. If the window can fit within the display at the location specified, it is moved to the new position. If a Change Window Coordinates exit routine is specified on the window description, it is called after the window is successfully moved. If the window is a full screen window, the API will complete successfully, but the window will not be moved.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window to be moved.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Resize Window (QsnRszWin) API

Required Parameter Group:

1 Window handle

Input Binary(4)

2 Number of rows

Input Binary(4)

3 Number of columns

Input Binary(4)

Omissible Parameter:

4 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Resize Window (QsnRszWin) API allows a window to be resized. If a Change Window Coordinates exit routine is specified on the window description, it is called after the window is successfully resized. If the window is a full screen window, the API will complete successfully, but the window will not be moved. Windows where the associated low-level environment indicates DBCS support cannot be resized.

Authorities and Locks

None

Required Parameter Group

Window handle
INPUT; BINARY(4)

A handle for the window to be resized.

Number of rows
INPUT; BINARY(4)

The new value for the number of rows in the window.

Number of columns
INPUT; BINARY(4)

The new value for the number of columns in the window.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A2 E	Window does not fit on screen.
CPFA3A4 E	Specified window is not active.
CPFA3A5 E	Window dimensions not within window limits.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Resize Window by User (QsnRszWinUsr) API

Required Parameter:

1 Window handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Resize Window by User (QsnRszWinUsr) API allows a window to be resized according to the cursor movements specified by the user. If the cursor is located on a border when this API is called, then that border can be moved. If the cursor is located on a border corner, the two sides that meet at that corner can be moved. If the user positions the cursor to a new row/column for the horizontal or vertical border, the border is moved to the new coordinate position and the window is resized accordingly. If the cursor is not on the border when the API is called, then the cursor is moved to the bottom center corner of the window and the user is prompted to move the cursor to the new position for the bottom center corner of the window. The prompt is displayed only if a message line has been defined. If the window is a full screen window, the API will complete successfully, but the window will not be moved.

A window can be made only as small (large) as the minimum (maximum) size allowed for the window. If the user moves the cursor such that the resulting window will be smaller (larger) than the minimum (maximum) size allowed, the resulting window will be the minimum (maximum) size. If a Change Window Coordinates exit routine is specified on the window description, this routine is called after the window is successfully resized. Typically, this API would be called after the user presses a particular function key. Windows where the associated low-level environment indicates DBCS support cannot be resized.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window to be resized.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Window Data (QsnRtvWinDta) API

Required Parameter:	
1	Window handle
Input	Binary(4)
Omissible Parameter Group:	
2	User data pointer
Output	PTR(SPP)
3	Error code
I/O	Char(*)
Returned Value:	
	User data pointer
Output	PTR(SPP)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Window Data (QsnRtvWinDta) API returns a pointer to the user data for the given window.

Authorities and Locks

None

Required Parameter

Window handle
INPUT; BINARY(4)

A handle for the window for which the user data should be returned.

Omissible Parameter Group

User data pointer
OUTPUT; PTR(SPP)

A space pointer to the user data field supplied in the user extension information when the window was defined. If no user data is associated with the window, the null pointer is returned.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

User data pointer

OUTPUT; PTR(SPP)

This API returns the value for the user data pointer parameter, or the null pointer if an error occurs.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1F E	Pointer is not on a 16 byte boundary.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Window Description (QsnRtvWinD) API

Required Parameter Group:

1 Window handle

Input Binary(4)

2 Window description

Output Char(*)

3 Length of window description

Input Binary(4)

Omissible Parameter:

4 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Window Description (QsnRtvWinD) API retrieves a copy of the window description for the given window.

Authorities and Locks

None

Required Parameter Group

Window handle
INPUT; BINARY(4)

A handle for the window for which the window description should be returned.

Window description
OUTPUT; CHAR(*)

The window description for the given window. The format of the data returned is shown in "Format of the Window Description Returned."

Length of window description
INPUT; BINARY(4)

The length of the window description parameter. If the length is larger than the size of the receiver variable, the results are not predictable. The minimum length is 8. The API returns as much information as it can fit in this length. If the available information is longer, it is truncated. If the available information is shorter, the unused output is unchanged; whatever is already stored in that space remains there. To determine how much information the API actually returns in response to this call, see the bytes returned field. To determine how much information the API could return if space were available, see the bytes available field.

Omissible Parameter

Error code
I/O; CHAR(*)

Thestructure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Window Description Returned

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(*)	Window description

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Window description. The format of the remaining data returned is shown in "Format of the Window Description" on page 180.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Set Window Services Attributes (QsnSetWinAtr) API

Required Parameter Group:

1 Window services attributes description

Input Char(*)

2 Length of window service attributes description

Input Binary(4)

Omissible Parameter:

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Set Window Services Attributes (QsnSetWinAtr) API sets the default attributes for the window services.

Authorities and Locks

None

Required Parameter Group

Window services attributes
INPUT; CHAR(*)

Defines the attributes for the window services APIs. The format of the window services attributes description is shown in "Format of the Window Services Attribute Description."

Length of window services attributes
INPUT; BINARY(4)

The length of the window services attributes parameter.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Window Services Attribute Description

Offset		Type	Field
Dec	Hex		
0	0	CHAR(3)	Monochrome window display attributes
3	3	CHAR(3)	Color window display attributes

Field Descriptions

Color window display attributes. The window display attributes for a color display. The first character is the attribute for the window border when the window is not current, the second for when the window is current, and the third for the leading window attribute. The first attribute is ignored for GUI windows, which only use the second attribute. All bytes can contain the same value. The special value X'00' indicates that no screen attribute is to be used for the given character. Both the current and noncurrent border attributes must be either X'00' or a valid attribute. For example, it is incorrect to specify the current attribute field X'00' and the noncurrent attribute field with a valid attribute.

The default values for these fields are X'20' for green, X'3A' for blue, and X'20' for green.

Monochrome window display attributes. The window display attributes for a monochrome display. The first character is the attribute for the window border when the window is not current, the second for when the window is current, and the third for the leading window attribute. The first attribute is ignored for GUI windows, which only use the second attribute. All bytes may contain the same value. The special value X'00' indicates that no screen attribute is to be used for the given character. Both the current and noncurrent border attributes must be either X'00' or a valid attribute. For example, it is incorrect to specify the current attribute field X'00' and the noncurrent attribute field with a valid attribute.

The default values for these fields are X'20' for normal attribute, X'22' for high intensity, and X'20' for normal attribute, respectively.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3AC E	Window services attributes description value is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Window I/O APIs

The majority of window I/O operations are performed through calls to the low-level services interfaces. If specified on the window description or through an explicit call to the Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API, the low-level interfaces can operate in a relative mode, where operations such as Set Field (QsnSetFld) are performed relative to the current window. See "Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API" on page 46 (QsnSetEnvWinMod) API for details.

The APIs specific to window I/O operations are:

- "Clear Window (QsnClrWin) API" on page 204 (QsnClrWin) clears the window area.
- "Clear Window Message (QsnClrWinMsg) API" on page 205 (QsnClrWinMsg) clears the message for a given window.
- "Display Window (QsnDspWin) API" on page 207 (QsnDspWin) draws the window border and clears the window area.
- "Put Window Message (QsnPutWinMsg) API" on page 209 (QsnPutWinMsg) puts a message on the message line for a given window.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Clear Window (QsnClrWin) API

Required Parameter:	
1	Window handle
Input	Binary(4)
Omissible Parameter:	
2	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Clear Window (QsnClrWin) API clears the window area for the given window. Any field definitions remain intact. Use the Clear Field Table (QsnClrFldTbl) API to remove field definitions.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window to be cleared.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Clear Window Message (QsnClrWinMsg) API

Required Parameter:

1 Window handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Clear Window Message (QsnClrWinMsg) API clears the window message for the given window. This API is valid only if the window has a message line specified for it.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window containing the message to be cleared.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3A7 E	Window does not have a message line.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Display Window (QsnDspWin) API

Required Parameter:		
1	Window handle	
Input	Binary(4)	
Omissible Parameter:		
2	Error code	
I/O	Char(*)	
Returned Value:		
	Return code	
Output	Binary(4)	
Default Public Authority: *USE		
Service Program: QSNAPI		
Threadsafe: No		

The Display Window (QsnDspWin) API draws the window border for the current window and clears the window area. The QsnDspWin API does not make a window current. It simply redraws the window using the existing border attributes. For overlapping windows, use this API only for the current window. If a Draw Window exit routine is specified on the window description, this routine is called after the window is defined successfully and prior to actually drawing the window.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window to be drawn.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Put Window Message (QsnPutWinMsg) API

Required Parameter:

1 Window handle

Input Binary(4)

Omissible Parameter Group:

2 Message

Input Char(*)

3 Message length

Input Binary(4)

4 Lock keyboard

Input Char(1)

5 Message identifier

Input Char(7)

6 Qualified message file name

Input Char(20)

7 Row

Input Binary(4)

8 Column

Input Binary(4)

9 Starting monochrome attribute

Input Char(1)

10 Ending monochrome attribute

Input Char(1)

11 Starting color attribute

Input Char(1)

12 Ending color attribute

Input Char(1)

13 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Put Window Message (QsnPutWinMsg) API places an error message on the message line for the given window. This API is valid only if the window has a message line specified for it.

Authorities and Locks

None

Required Parameter

Window handle
INPUT; BINARY(4)

A handle for the window in which the message should be placed.

Omissible Parameter Group

Message
INPUT; CHAR(*)

The message to be displayed. If the message does not fit within the window, it is truncated to fit. If the message length parameter is specified as nonzero, the message parameter is required. The message or the message ID parameter must be specified. If the message parameter is specified, the message ID parameter is ignored and no help key support is available for the message.

Message length
INPUT; BINARY(4)

The number of bytes of message data to be displayed.

Lock keyboard
INPUT; CHAR(1)

Whether the keyboard should be placed in prehelp error state of not. The possible values are:

0	Do not place the keyboard in prehelp error state.
1	Place the keyboard in prehelp error state. If 1 is specified, the processing of this API follows that of the QsnSetErr API and the QsnPutWinMsg API must be followed by an AID-associated read API. This is the default.

Message identifier
INPUT; CHAR(7)

The identifying code for the predefined message to be displayed. The first level text is displayed. If the user moves the cursor to the message line and presses the Help key, the message `No help text available` is displayed. This parameter is required if the message parameter is omitted.

Qualified message file name
INPUT; CHAR(20)

The name of the message file from which to retrieve the message information, and the library in which the message file resides. This parameter is required if the message parameter is omitted. The format of this parameter is:

Bytes	Value
1-10	Message file name
11-20	Message file library. This can be an actual library name or one of the special values *CURLIB or *LIBL.

Row INPUT; BINARY(4)

The relative window row at which to position the cursor when the message is displayed. To move the cursor, the API must be followed by an AID-associated read API.

If both row and column are omitted or specified with a zero value, the cursor is not moved. Row and column must both be specified or omitted; one cannot be specified if the other is omitted.

Column
INPUT; BINARY(4)

The relative window column at which to position the cursor when the message is displayed.

Starting monochrome attribute
INPUT; CHAR(1)

The initial screen attribute for monochrome displays. If this parameter is omitted and monochrome attributes are to be used, no initial attribute is written to the display for the data.

The monochrome and color attributes parameters are the initial and ending screen attributes: an initial and ending screen attribute to be used for a monochrome or a color display, respectively. One of these parameters will be selected based on the underlying display type, and the other will be discarded. Any of the attributes can be specified as a special value, X'00', indicating that no screen attribute should be written to the display. If the initial screen attribute is specified as an actual attribute, the data column, if specified, must be greater than or equal to 2. The initial screen attribute, if not X'00', will be written to the screen at the column previous to the first data character if row and column are specified, otherwise to the current display address. The ending screen attribute, if not X'00', will be written at the column directly after the last data character.

See Screen Attribute Characters for a description of the screen attribute values.

Ending monochrome attribute
INPUT; CHAR(1)

The ending screen attribute for monochrome displays. If this parameter is omitted and monochrome attributes are to be used, no ending attribute is written to the display for the data.

Starting color attribute
INPUT; CHAR(1)

The initial screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no initial attribute is written to the display for the data. See Screen Attribute Characters for a description of the screen attribute values.

Ending color attribute
INPUT; CHAR(1)

The ending screen attribute for color displays. If this parameter is omitted and color attributes are to be used, no ending attributes are written to the display for the data.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA307 E	Screen position &1, &2 outside of display or window area.
CPFA30D E	Invalid screen attribute.
CPFA31E E	Required parameter &1 omitted.
CPFA333 E	Parameter &1 not positive integer value.
CPFA335 E	Screen address parameter error.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3A7 E	Window does not have a message line.
CPFA3A8 E	Error occurred retrieving message text.
CPFA3AA E	Window handle incorrect.
CPFA3AB E	The value for &1 must be '0' or '1'.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Window Manager Services APIs

The window manager services APIs manage multiple windows, support I/O to several active windows, and allow switching between windows. Much of the work of the window manager services is performed implicitly through the window builder routines.

Windows are managed on an activation-group basis. That is, all windows that were started within a given activation group will be managed as a unit. You can only switch to or end a window that was started within the current activation group. If windows need to be redrawn, only those windows within the current activation group will be redrawn. A window can be created in one activation group and started in another group. The activation group in which the window was started will be the group to manage the window.

The window manager services APIs are:

- "End a Window (QsnEndWin) API" on page 213 (QsnEndWin) ends an active, current window and removes it from the screen.
- "Retrieve Current Window (QsnRtvCurWin) API" on page 214 (QsnRtvCurWin) returns the handle for the current window.
- "Set Current Window (QsnSetCurWin) API" on page 216 (QsnSetCurWin) makes the specified window current.
- "Start a Window (QsnStrWin) API" on page 217 (QsnStrWin) starts a window by displaying it and making it the current window.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

End a Window (QsnEndWin) API

Required Parameter:

1 Window handle

Input Binary(4)

Omissible Parameter Group:

2 Restore screen

Input Char(1)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The End a Window (QsnEndWin) API ends a currently active window that was started with the Start a Window (QsnStrWin) API. The window is removed from the display on the screen and from the active window list. The data associated with the window is not deallocated.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window to be ended.

Omissible Parameter Group

Restore screen

INPUT; CHAR(1)

Indicates if the underlying display image should be restored when the window is ended. This parameter is ignored if the underlying display image was not saved. This option should be used if the screen will be refreshed by another application and does not need to be refreshed when the window is removed. Performance can be improved by not restoring the display image. However, the saved screen may not be restored properly if it is not restored by another application.

The possible values are:

0	Do not restore the screen when the window is ended.
1	Restore the screen, if saved, when the window is ended. This is the default.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.
CPFA3AB E	The value for &1 must be '0' or '1'.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Current Window (QsnRtvCurWin) API

Omissible Parameter Group:

1 Current window handle

Output Binary(4)

2 Error code

I/O Char(*)

Returned Value:

Current window handle

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Current Window (QsnRtvCurWin) API returns the handle for the current window.

Authorities and Locks

None

Omissible Parameter Group

Current window handle
OUTPUT; BINARY(4)

The variable that contains the handle for the current window when the QsnRtvCurWin API has completed. If there is no current window, this parameter is set to 0.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Current window handle
OUTPUT; BINARY(4)

This API returns the value for the current window handle parameter. If there is no current window, this API returns 0.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Set Current Window (QsnSetCurWin) API

Required Parameter:	
1	Window handle
Input	Binary(4)
Omissible Parameter:	
2	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Set Current Window (QsnSetCurWin) API makes the given window the current window. The QsnSetCurWin API draws the window with the current window border attribute, if specified. The Current Window exit routine, if specified on the window description, is called after the given window becomes current. The current window overlays all other windows on the display screen.

Authorities and Locks

None

Required Parameter

Window handle

INPUT; BINARY(4)

A handle for the window that will become current.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3AA E	Window handle incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Start a Window (QsnStrWin) API

Required Parameter:

1 Window handle

Input Binary(4)

Omissible Parameter Group:

2 Save screen

Input Char(1)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Start a Window (QsnStrWin) API starts a window created with the Create a Window (QsnCrtWin) API. This causes the window to be displayed on the screen and added to the active window list. If specified, the Draw Window exit routine is called immediately before the window is drawn.

Authorities and Locks

None

Required Parameter

Window handle
INPUT; BINARY(4)

A handle for the window to be started.

Omissible Parameter Group

Save screen
INPUT; CHAR(1)

Indicates if the underlying display image should be saved prior to drawing the window. This option should be used only if the window will not be moved or resized over an existing display image. Performance can be improved by not saving the display image. However, doing this limits the overlapping nature of the window. If an attempt is made to move or resize a window for which the display image was not saved, the screen is cleared and all windows are redrawn prior to moving the window.

The possible values for this parameter are:

0	Do not save the underlying display image when the window is started.
1	Save the underlying display image when the window is started. This is the default.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3AA E	Window handle incorrect.
CPFA3AB E	The value for &1 must be '0' or '1'.

Performance Considerations

You can improve the performance of window operations by doing the following:

- Do not save or restore the underlying screen image when a window is started or ended with the Start a Window (QsnStrWin) or End a Window (QsnEndWin) API, respectively. See pages "Start a Window (QsnStrWin) API" on page 217 and "End a Window (QsnEndWin) API" on page 213.
- For non-GUI windows, use the same color for current and noncurrent boundaries.
- Use a display station attached to a control unit that supports an enhanced interface for a nonprogrammable work station, even if you are not using GUI windows.
- Use GUI window support when the underlying control unit supports this.

Creating/Manipulating Windows Example

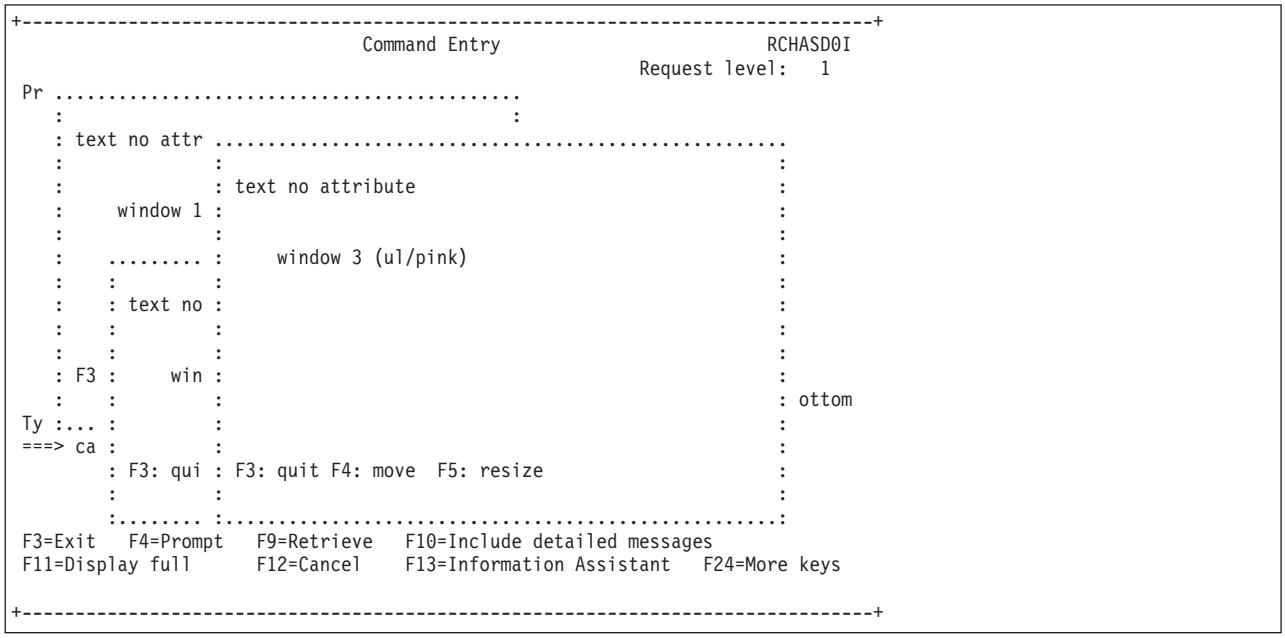
See Code disclaimer information for information pertaining to code examples.

The sample program in Creating and Manipulating Windows (page 219) shows how to create and manipulate several windows with exit routines. The program creates three windows- Window 1, Window 2, and Window 3. Each time Enter is pressed, the next window is made current; in which case, the Draw Window exit routine for that window is called. If the user presses F4=Move or F5=Resize, the current window is moved or resized and the Draw Window exit routine is called again. The resulting screen output is shown in Display Screen (page 221).

Creating and Manipulating Windows

```
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "qsnapi.h"


void GenericDraw(const Qsn_Cmd_Buf_T *cbuf, const Qsn_Win_T *win)
{
    char *msg1 = "F3: quit F4: move  F5: resize";
    char *msg2 = "text no attribute";

    QsnWrtDta(msg2, strlen(msg2), 0, 2, 1, QSN_NO_SA, QSN_NO_SA,
              QSN_NO_SA, QSN_NO_SA, *cbuf, *win, NULL);
    QsnWrtDta(msg1, strlen(msg1), 0, -1, 1, QSN_SA_HI, QSN_SA_NORM,
              QSN_SA_RED, QSN_SA_NORM, *cbuf, *win, NULL);
}

void Draw1(const Qsn_Win_T *win, const Qsn_Cmd_Buf_T *cbuf)
{
    char *txt = "window 1 (ul/blue)";

    GenericDraw(cbuf, win);
    QsnWrtDta(txt, strlen(txt), 0, 5, 5, QSN_SA_UL, QSN_SA_NORM,
              QSN_SA_BLU, QSN_SA_NORM, *cbuf, *win, NULL);
}

void Draw2(const Qsn_Win_T *win, const Qsn_Cmd_Buf_T *cbuf)
{
    char *txt = "window 2 (ul/red)";

    GenericDraw(cbuf, win);
    QsnWrtDta(txt, strlen(txt), 0, 5, 5, QSN_SA_UL, QSN_SA_NORM,
              QSN_SA_RED, QSN_SA_NORM, *cbuf, *win, NULL);
}

void Draw3(const Qsn_Win_T *win, const Qsn_Cmd_Buf_T *cbuf)
{
    char *txt = "window 3 (ul/pink)";

    GenericDraw(cbuf, win);
    QsnWrtDta(txt, strlen(txt), 0, 5, 5, QSN_SA_UL, QSN_SA_NORM,
              QSN_SA_PNK, QSN_SA_NORM, *cbuf, *win, NULL);
}
int main (void) {
    int i;
    char text[100];
    Qsn_Win_T win1, win2, win3, cur;
    Qsn_Win_Desc_T  win_desc;
    Qsn_Win_Ext_Inf_T ext = { NULL, NULL, NULL, NULL, NULL, NULL };
    Q_Bin4          win_desc_length = sizeof(win_desc);
    char aid;

    QsnInzWinD(&win_desc, win_desc_length, NULL);
    win_desc.GUI_support = '0';

    /* define and start window 1 */
    win_desc.top_row = 3;
    win_desc.left_col = 5;
    win_desc.num_rows = 13;
    win_desc.num_cols = 40;
    ext.draw_fp = Draw1;
    win1 = QsnCrtWin(&win_desc, win_desc_length, &ext, sizeof(ext),
                     '1', NULL, 0, NULL, NULL);
    QsnGetAID(NULL, 0, NULL);

    /* define and start window 2 */
    win_desc.top_row = 10;
    win_desc.left_col = 10;
    win_desc.num_rows = 10;
    win_desc.num_cols = 30;
    ext.draw_fp = Draw2;
    win2 = QsnCrtWin(&win_desc, win_desc_length, &ext, sizeof(ext),
                     '1', NULL, 0, NULL, NULL);
    QsnGetAID(NULL, 0, NULL);

    /* define and start window 3 */
    win_desc.top_row = 5;
    win_desc.left_col = 20;
    win_desc.num_rows = 15;
    win_desc.num_cols = 50;
    ext.draw_fp = Draw3;
```

Display Screen



API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Session Services APIs

The session services APIs provide a general scrolling I/O interface. They can be used to build a standard input-line scrolling interface or an interface that has an output-only scroll area (called a **scroller**) in a window. Sessions are special cases of windows as supported by the window services. A session is defined using a session, a window, and a low-level environment description. The window and low-level environment descriptions are the same as those used to define a window directly with the window services APIs. The session description defines the structure of the session. The structure includes the coordinates of the scrolling portion, the length of the input line, the amount to roll by, and so on. A session is implemented as a window, where the window user data pointer describes the session itself. Thus, a session can be manipulated through the window and low-level interfaces by passing the session handler or through the session interfaces. This implementation is similar to the concept of inheritance in object-oriented programming languages.

Sessions are similar in concept to subfiles and can be used for any application that requires a scrolling line interface. The session services APIs are divided into the following functional groups:

- "Session Manipulation and Query APIs" on page 222 allow you to create, query, and manipulate sessions.
- "Session I/O APIs" on page 258 allow you to perform input and output operations to sessions.

For additional information, select one of the following:

- "Session Details" on page 301
- "Line Mode and Character Mode I/O" on page 302
- "Command Key Action Routines" on page 302

- "Action Routine Parameters" on page 303
- "Active Position" on page 304
- "EBCDIC Display Control Characters" on page 304
- "DBCS Considerations" on page 305

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Session Manipulation and Query APIs

The session manipulation and query APIs are:

- "Change Session (QsnChgSsn) API" on page 223 (QsnChgSsn) changes the description for a session.
- "Clear Scroller (QsnClrScl) API" on page 225 (QsnClrScl) clears the scroller data.
- "Create a Session (QsnCrtSsn) API" on page 227 (QsnCrtSsn) creates a session for subsequent session I/O operations.
- "Display Scroller Bottom (QsnDspSclB) API" on page 238 (QsnDspSclB) shows the last line of scroller data.
- "Display Scroller top (QsnDspSclT) API" on page 239 (QsnDspSclT) shows the first line of scroller data.
- "Initialize Session Description (QsnInzSsnD) API" on page 241 (QsnInzSsnD) initializes a session description with default values.
- "Query If Scroller in Line Wrap Mode (QsnQrySclWrp) API" on page 242 (QsnQrySclWrp) queries if line wrap mode is on or off.
- "Retrieve Number of Columns to Shift Scroller (QsnRtvSclNumShf) API" on page 244 (QsnRtvSclNumShf) returns number of columns to shift scroller by.
- "Retrieve Number of Rows to Roll Scroller (QsnRtvSclNumRoll) API" on page 245 (QsnRtvSclNumRoll) returns the number of rows to roll scroller by.
- "Retrieve Session Data (QsnRtvSsnDta) API" on page 247 (QsnRtvSsnDta) returns a pointer to the user data for a session.
- "Retrieve Session Description (QsnRtvSsnD) API" on page 248 (QsnRtvSsnD) retrieves a copy of the description for a session.
- "Roll Scroller Down (QsnRollSclDown) API" on page 250 (QsnRollSclDown) rolls the scroller down.
- "Roll Scroller Up (QsnRollSclUp) API" on page 252 (QsnRollSclUp) rolls the scroller up.
- "Shift Scroller left (QsnShfSclL) API" on page 253 (QsnShfSclL) shifts the scroller to the left.
- "Shift Scroller Right (QsnShfSclR) API" on page 255 (QsnShfSclR) shifts the scroller to the right.
- "Toggle Line Wrap/Truncate Mode (QsnTglSclWrp) API" on page 257 (QsnTglSclWrp) toggles the session between line wrap and truncation mode.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Change Session (QsnChgSsn) API

Required Parameter Group:	
1	Session handle
Input	Binary(4)
2	Session description
Input	Char(*)
3	Length of session description
Input	Binary(4)
Omissible Parameter:	
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Change Session (QsnChgSsn) API changes the session description for the given session. If the session is currently displayed, it will be redrawn to reflect any changes.

Authorities and Locks

Exit Routine Authority
*EXECUTE

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session for which the session description is to be changed.

Session description
INPUT; CHAR(*)

The format of the session description is shown in "Format of the Session Description" on page 229.

Length of session description
INPUT; BINARY(4)

The length of the session description parameter.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape essages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA314 E	Memory allocation error.
CPFA318 E	Error calling exit routine.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D1 E	Session description value is incorrect.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Clear Scroller (QsnClrScl) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Resize indication

Input Char(1)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Clear Scroller (QsnClrScl) API clears the scroller data associated with a session and optionally resizes the scroller buffer.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be cleared. All data in the scroller will be cleared.

Omissible Parameter Group

Resize indication

Input; CHAR(1)

Whether the scroller buffer should be resized when it is cleared.

0	Maintain current buffer size and data. This is the default.
1	Resize the buffer to the initial size given on the session description.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA314 E	Memory allocation error.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3AB E	The value for &1 must be '0' or '1'.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Create a Session (QsnCrtSsn) API

Required Parameter Group:

1	Session description
Input	Char(*)
2	Length of session description
Input	Binary(4)

Omissible Parameter Group:

3	User extension information
Input	Char(*)
4	Length of user extension information
Input	Binary(4)
5	Start session flag
Input	Char(1)
6	Window description
Input	Char(*)
7	Length of window description
Input	Binary(4)
8	Low-level environment description
Input	Char(*)
9	Length of low-level environment description
Input	Binary(4)
10	Session handle
Output	Binary(4)
11	Error code
I/O	Char(*)

Returned Value:

	Session handle
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Create a Session (QsnCrtSsn) API creates a session and returns a handle for the created session. The session must be deleted using the Delete Low-Level Environment (QsnDltEnv) API.

Authorities and Locks

None.

Required Parameter Group

Session description
INPUT; CHAR(*)

The defined attributes of the session to be created. It must be declared aligned on a 16-byte boundary. The format of the session description is shown in "Format of the Session Description" on page 229.

Length of session description
INPUT; BINARY(4)

The length of the session description parameter.

Omissible Parameter Group

User extension information
INPUT; CHAR(*)

Information that is used to associate data and exit routines with the session. This parameter is required if the user extension information length parameter is supplied. This essentially enables the object-oriented programming concept of inheritance, allowing the session to be extended in a natural way. The user extension data cannot be changed once the session has been created. The format of this parameter is shown in the section "Format of the Session User Extension Data" on page 233.

Length of user extension information
INPUT; BINARY(4)

The length of the user extension information parameter.

Start session flag
INPUT; CHAR(1)

Whether or not the session should be displayed on screen when it is created. The possible values are:

0	Do not display the session on the screen when it is created. If you specify this value, you must use the Start a Window (QsnStrWin) API to display the session.
1	Display the session on the screen when it is created. This is the default.

Window description
INPUT; CHAR(*)

The defined attributes for the window containing the session. This parameter is required if the window description length parameter is supplied. The format of the window description is shown in "Format of the Window Description" on page 180. If this parameter is omitted, a window will be created with default values.

Length of window description
INPUT; BINARY(4)

The length of the window description parameter.

Low-level environment description
INPUT; CHAR(*)

operating environment for low-level operations used to create and manipulate the windows. This parameter is required if the low-level environment description length parameter is supplied. The

format of the low-level environment description is shown in “Format of the Low-Level Environment Description” on page 10. If this parameter is omitted, a low-level environment will be created with default values.

Length of low-level environment description
INPUT; BINARY(4)

The length of the low-level environment description parameter.

Session handle
OUTPUT; BIN(4)

The variable containing a handle for the created session after the QsnCrtSsn API has completed. This handle can be used across activation groups if the activation group in which the handle was created is still active.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Session handle
OUTPUT; BINARY(4)

This API returns the value for the session handle parameter or -1 if an error occurs during processing.

Format of the Session Description

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	PTR(PP) [24]	Array of command key actions
384	180	BINARY(4)	top row of scroller
388	184	BINARY(4)	left column of scroller
392	188	BINARY(4)	Number of rows in scroller
396	18C	BINARY(4)	Number of columns in scroller
400	190	BINARY(4)	Default number of rows to roll scroller by
404	194	BINARY(4)	Default number of columns to shift scroller by
408	198	BINARY(4)	Scroller buffer initial size
412	19C	BINARY(4)	Scroller buffer maximum size
416	1A0	BINARY(4)	Scroller buffer increment
420	1A4	BINARY(4)	Number of rows for input line
424	1A8	CHAR(1)	Reserved
425	1A9	CHAR(1)	Wrap indication
426	1AA	CHAR(1)	Reserved
427	1AB	CHAR(1)	Display control characters indication
428	1AC	CHAR(1)	Echo session input
429	1AD	CHAR(1)	Show scroller lines
430	1AE	CHAR(1)	Show scroller characters
431	1AF	CHAR(1)	Show command key descriptions
432	1B0	CHAR(1)	Command key attribute for a monochrome display

Offset		Type	Field
Dec	Hex		
433	1B1	CHAR(1)	Command key attribute for a color display
434	1B2	CHAR(1)	Input line attribute for a monochrome display
435	1B3	CHAR(1)	Input line attribute for a color display
436	1B4	BINARY(4)	Offset to input line prompt
440	1B8	BINARY(4)	Length of input line prompt
444	1BC	BINARY(4)	Offset to command key description line 1
448	1C0	BINARY(4)	Length of command key description line 1
452	1C4	BINARY(4)	Offset to command key description line 2
456	1C8	BINARY(4)	Length of command key description line 2
460	1CC	CHAR(20)	Reserved.
*	*	CHAR(*)	Input line prompt
*	*	CHAR(*)	Command key description line 1
*	*	CHAR(*)	Command key description line 2

Field Descriptions

In the following descriptions, the default value refers to the value set by the Initialize Session Description (QsnInzSsnD) API.

Array of command key actions. An array of 24 function pointers, each corresponding to the action to be performed when the associated command key is pressed. An element that is specified as a null pointer indicates that the command key is invalid. An element can also be set to the dummy routine QsnSameAction, in which case the current action routine for that key is used. If a command key action is set to the dummy routine QsnDefaultAction, then the default action for that key is used. The defaults for command key actions and the parameters passed to the action routines are described in "Command Key Action Routines" on page 302. The procedures are exported as part of the service program that contains the DSM session services.

Command key attribute for a color display. The default value is X'28' for red.

Command key attribute for a monochrome display. The default value is X'20' for normal attribute.

Command key description line 1. The text string for the first line of command key descriptions.

Command key description line 2. The text string for the second line of command key descriptions.

Default number of columns to shift scroller by. The default number of columns to shift scroller by for the Shift Scroller left (QsnShfSclL) and Shift Scroller center (QsnShfSclR) APIs. This value must be a positive integer value. If 0 is specified, the default is the number of columns in the scroller less two (two scroller columns are reserved for the prefix area).

Default number of rows to roll scroller by. The default number of rows to roll scroller by for the Roll Scroller Up (QsnRollSclUp) and Roll Scroller Down (QsnRollSclDown) APIs. This value must be a positive integer value. If 0 is specified, the default is the number of rows in the scroller.

Display control characters indication. Specifies whether or not scroller lines contain EBCDIC display control characters. If the data contains such control characters and this is not indicated, unexpected results can occur. (See "EBCDIC Display Control Characters" on page 304 for details of the control characters supported and their interpretation.) The possible values are:

0	Scroller lines do not contain EBCDIC display control characters. This is the default when the display device or emulator does not support DBCS data.
1	Scroller lines contain EBCDIC display control characters. If 1 is specified, any data written to the session with a value below X'40' is converted to blank. This is the default when the display device or emulator supports DBCS data.

Echo session input. Specifies whether lines entered at the session command line are to be echoed to the scroller. The possible values are:

0	Do not echo session input lines to the scroller.
1	Echo session input lines to the scroller. This is the default.

Input line attribute for a color display. The default value is X'24' for green underline attribute. If X'00' is specified, the default value is used.

Input line attribute for a monochrome display. The default value is X'24' for underline attribute. If X'00' is specified, the default value is used.

Input line prompt. The text string for the input line prompt.

Left column of scroller. This position is relative to the left of the window which is column 1. The default is 1.

Length of command key description line 1. This value must not exceed the maximum number of columns in the window. The default value is 0. No space is used in the session for this line. If the description line cannot be displayed completely within the window, it is truncated to fit.

Length of command key description line 2. This value must not exceed the maximum number of columns in the window. The default value is 0. No space is used in the session for this line. If the description cannot be displayed completely in the window, it is truncated to fit.

Length of input line prompt. This value must not exceed the maximum number of columns in the window. A value of 0 specifies that there is no prompt. The default value is -1 and corresponds to the default input line prompt ===.

If the input line cannot be displayed completely within the window, it is truncated to fit. The input line will continue on the next window line.

Number of columns in scroller. This value must be a positive integer no greater than the number of columns in the session window. If 0 is specified, the default is the number of columns remaining in the window from the left column of the scroller. This value includes the 2 bytes used for the prefix area to the left of the scroller input line.

If the low level environment supports DBCS, this value must be greater than or equal to 6 to allow for the size of the prefix area, the shift control (SO/SI) characters and one DBCS character.

If the display device or emulator supports CCSID-based I/O, this value must be greater than or equal to 4 to allow for the size of the prefix area and one UCS2 character.

Number of rows for input line. The input line starts in the row specified by the formula: last window row less the number of rows required for input line less the number of rows required for the function key descriptions. The input line will start one column past the end of the input line prompt. If there is no

input line prompt, then the input line starts one byte to the center of the leftmost usable column of the window. If this value is 0, then no input line is created. The default is 1.

Number of rows in scroller. This value must be a positive integer no greater than the number of rows in the session window. If 0 is specified, the default is the number of rows remaining in the window from the top row of the scroller.

Offset to command key description line 1. The offset from the beginning of the structure to the start of the command key description line 1. This field is ignored if the length of command key description line 1 field specifies no command key description. The offset plus the length must be less than the session description length.

Offset to command key description line 2. The offset from the beginning of the structure to the start of the command key description line 2. This field is ignored if the length of command key description line 2 field specifies no command key description. The offset plus the length must be less than the session description length.

Offset to input line prompt. The offset from the beginning of the structure to the start of the input line prompt. This field is ignored if the length of input line prompt field specifies no prompt or the default prompt. The offset plus the length must be less than the session description length.

Reserved. This field must be set to X'00'.

Scroller buffer increment. Specifies, in bytes, the amount to increment the scroller buffer size by when the buffer is full and the buffer-full action is to increment the buffer size. The default value is 2000 bytes.

If the scroller buffer cannot be incremented because of insufficient resources, data at the beginning of the scroller will be removed to create space for the new data.

Scroller buffer initial size. The initial buffer size, in number of bytes, that will be allocated for storing the session scroller lines. The default value is 2000 bytes.

Scroller buffer maximum size. The maximum buffer size, in bytes, that will be allocated for storing the session scroller lines. The default value is 0, indicating no maximum size.

Show command key descriptions. Whether or not the function key description lines are to be shown. The possible values are:

0	Do not show function key descriptions.
1	Show function key descriptions. This is the default.

Show scroller characters. Whether or not characters written to the scroller in character mode are shown immediately on the screen. You can use the scroller line and character display options together to specify that groups of characters are not displayed immediately, but each complete line is. The possible values are:

0	Do not show characters on the screen as they are written. Use the Display Scroller bottom (QsnDspSclB) API to display the scroller data on the screen. This is the default.
1	Show scroller characters on the screen as they are written.

Show scroller lines. Whether or not lines written to the scroller are shown immediately on the screen. The possible values are:

0 Do not show scroller lines on the screen as they are written. Use the Display Scroller bottom (QsnDspSclB) API to display the scroller lines on the screen. This is the default.

1 Show scroller lines on the screen as they are written.

Top row of scroller. This position is relative to the top of the window, which is row 1. The default is 1.

Wrap indication. How to handle lines that do not fit within the session window. Possible values are:

0 Truncate lines that do not fit. The truncated portion of the line may be viewed by scrolling to the center.

1 Wrap lines to the next line. This is the default.

Format of the Session User Extension Data

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	PTR(SPP)	User data associated with the session
16	10	PTR(PP)	Exit routine to call when the session is changed
32	20	PTR(PP)	Exit routine to call when window is deleted
48	30	PTR(PP)	Exit routine to call when window coordinates are changed
64	40	PTR(PP)	Exit routine to call when window is drawn
80	50	PTR(PP)	Exit routine to call when this window made current

Field Descriptions

Exit routine to call when session changed. The exit routine to call when a session is changed using the Change Session (QsnChgSsn) API.

Exit routine to call when window coordinates changed. The exit routine to call when the window coordinates are changed using the QsnMovWin, Move Window by User (QsnMovWinUsr), Resize Window (QsnRszWin), or Resize Window by User (QsnRszWinUsr), APIs.

Exit routine to call when window deleted. The exit routine to call when a window is deleted using the Delete Low-Level Environment (QsnDltEnv) API.

Exit routine to call when window drawn. The exit routine to call when a window is drawn using the Display Window (QsnDspWin) API.

Exit routine to call when window made current. The exit routine to call when this window is made current using the Set Current Window (QsnSetCurWin) API.

User data associated with the session. This is a pointer to any data that the user wants to associate with this session.

Session Exit Routines

Exit routines are user-supplied functions with a defined interface. The routines are called from certain APIs and allow the programmer to attach additional function to those APIs. For instance, if fields have been set up in a window, a Change Coordinates exit routine could be supplied to move the fields if the window is moved.

Change Session Exit Routine

This exit routine, if specified on the user extension information, is called when the session is changed. The following parameter is passed to the exit routine:

Parameter Passed to Exit Routine	
1	Session handle
Input	Binary(4)

Change Session Exit Routine Parameter

Session handle
INPUT; BINARY(4)

The session that was changed.

Delete Session Exit Routine

This exit routine, if specified on the user extension information, is called when the session is deleted. The following parameter is passed to the exit routine:

Parameter Passed to Exit Routine	
1	Session handle
Input	Binary(4)

Delete Session Exit Routine Parameter

Session handle
INPUT; BINARY(4)

The session that was deleted.

Change Session Coordinates Exit Routine

This exit routine, if specified on the user extension information, is called when the move or resize APIs are called. It is called after the session has been successfully moved or resized, but before the session is drawn on the screen. For this reason, you should not use this exit routine to draw anything in the session. The draw exit routine will be called when the session is moved or resized. The following parameters are passed to the exit routine:

Parameters Passed to Exit Routine	
1	Session handle
Input	Binary(4)
2	Top border offset
Input	Binary(4)
3	Left border offset
Input	Binary(4)
4	Bottom border offset
Input	Binary(4)
5	Center border offset
Input	Binary(4)

Change Session Coordinates Exit Routine Parameters

Session handle
INPUT; BINARY(4)

The Session for which the coordinates were changed.

Top border offset
INPUT; BINARY(4)

The offset, in screen rows, from the previous top session border to the current top session border (after the session coordinates have been changed). It can be positive, negative, or zero, depending on how the top session border was changed. For example, if the top border was moved down two rows, this value would be 2; if it was moved up 4 rows, this value would be -4; if the top row was not changed, this value would be 0.

Left border offset
INPUT; BINARY(4)

The offset, in screen columns, from the previous left session border to the current left session border (after the session coordinates have been changed). It can be positive, negative, or zero, depending on how the left session border was changed. For example, if the left border was moved two columns to the center, this value would be 2; if it was moved 4 columns to the left, this value would be -4, and if the left column was not changed, this value would be 0.

Bottom border offset
INPUT; BINARY(4)

The offset, in screen rows, from the previous bottom session border to the current bottom session border (after the session coordinates have been changed). It can be positive, negative, or zero, depending on how the bottom session border was changed. For example, if the border was moved down two rows, this value would be 2; if it was moved up 4 rows, this value would be -4; if the bottom row was not changed, this value would be 0.

Center border offset
INPUT; BINARY(4)

The offset, in screen columns, from the previous center session border to the current center session border (after the session coordinates have been changed). It can be positive, negative, or zero, depending on how the center session border was changed. For example, if the center border

was moved two columns to the center, this value would be 2; if it was moved 4 columns to the left, this value would be -4; if the center column was not changed, this value would be 0.

Exit Routine Error Handling

If an exception occurs during the processing of an exit routine, the exception is ignored and processing continues. A CPFA318 will be issued as a diagnostic message only. You can explicitly handle errors in an exit routine by adding an exception handler to the routine.

Draw Session Exit Routine

This exit routine, if specified on the user extension information, is called when the Display Window (QsnDspWin) API is called, before the session is drawn. Because the exit routine is called before the session is drawn, you should only write inside the session using the command buffer parameter. Direct operations should not be used for the exit routine.

The following parameters are passed to the exit routine:

Parameters Passed to Exit Routine	
1	Session handle
Input	Binary(4)
2	Command buffer
Input	Binary(4)

Draw Session Exit Routine Parameters

Session handle
INPUT; BINARY(4)

The session to be drawn.

Command buffer
INPUT; BINARY(4)

The command buffer used to store the commands that re-create the window contents. The contents of the command buffer are written to the screen along with the window border. This allows the window and its contents to be redrawn in a single I/O operation.

Current Session Exit Routine

This exit routine, if specified on the user extension information, is called when the session is made current through the Set Current Window (QsnSetCurWin) API. The following parameter is passed to the exit routine:

Parameter Passed to Exit Routine	
1	Session handle
Input	Binary(4)

Current Session Exit Routine Parameter

Session handle
INPUT; BINARY(4)

A handle for the session that is made current.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA314 E	Memory allocation error.
CPFA318 E	Error calling exit routine.
CPFA327 E	Low level environment description value incorrect.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A1 E	Window description value is incorrect.
CPFA3AB E	The value for &1 must be '0' or '1'.
CPFA3D1 E	Session description value is incorrect.

Additional errors may be generated by this API. They are listed under the applicable API as follows:

Error Category (API)	Page Reference
Environment description (QsnCrtEnv)	"Error Messages" on page 15
Window description (QsnCrtWin)	Error Messages

For examples of Create Session APIs, see Create Session and Read Data—Example.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Display Scroller Bottom (QsnDspSclB) API

Required Parameter:	
1	Session handle
Input	Binary(4)
Omissible Parameter:	
2	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Display Scroller Bottom (QsnDspSclB) API positions the scroller at the last line in the scroller area. As many lines preceding the last line as can fit in the scroller area are displayed as well.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be manipulated.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Display Scroller top (QsnDspSclT) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Display Scroller top (QsnDspSclT) API positions the scroller at the first line in the scroller area. As many lines following the first line as can fit in the scroller area are displayed as well.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be manipulated.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Initialize Session Description (QsnInzSsnD) API

Required Parameter Group:	
1	Session description
Output	Char(*)
2	Length of session description
Input	Binary(4)
Omissible Parameter:	
3	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Initialize Session Description (QsnInzSsnD) API initializes a session description with default values. Unless otherwise specified in the session description (see "Format of the Session Description" on page 229), pointer fields are set to the null pointer, numeric fields to 0, character flag fields to 0, and other character fields to blanks. For example, the default value for the wrap indication is 1, so this field will be set to 1.

Authorities and Locks

Exit Routine Authority
*EXECUTE

Required Parameter Group

Session description
OUTPUT; CHAR(*)

The session description to be initialized.

Length of session description
INPUT; BINARY(4)

The length of the session description parameter.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Query If Scroller in Line Wrap Mode (QsnQrySclWrp) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Wrap indication

Output Char(1)

3 Error code

I/O Char(*)

Returned Value:

Wrap indication

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Query If Scroller in Line Wrap Mode (QsnQrySclWrp) API queries if line wrap mode is set on or off for the given session.

Authorities and Locks

None

Required Parameter

Session handle
INPUT; BINARY(4)

A handle for the session to be queried.

Omissible Parameter Group

Wrap indication
OUTPUT; CHAR(1)

Whether line wrap mode is on or off. The possible values are:

0	Line wrap mode is off.
1	Line wrap mode is on.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Wrap indication
OUTPUT; BINARY(4)

This API returns the value for the wrap indication parameter if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Number of Columns to Shift Scroller (QsnRtvSclNumShf) API

Required Parameter:	
1	Session handle
Input	Binary(4)
Omissible Parameter Group:	
2	Shift amount
Output	Binary(4)
3	Error code
I/O	Char(*)
Returned Value:	
	Shift amount
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Number of Columns to Shift Scroller (QsnRtvSclNumShf) API returns the default number of columns to shift the scroller by for the Shift Scroller Left (QsnShfSclL) and Shift Scroller Right (QsnShfSclR) APIs. The default number of columns is specified on the session description. See "Create a Session (QsnCrtSsn) API" on page 227 and "Change Session (QsnChgSsn) API" on page 223 for details.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be queried.

Omissible Parameter Group

Shift amount

OUTPUT; BINARY(4)

The variable that contains the number of scroller columns to shift by when the QsnRtvSclNumShf API has completed.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Shift amount
OUTPUT; BINARY(4)

Returns the value for the shift amount parameter if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Number of Rows to Roll Scroller (QsnRtvSclNumRoll) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Roll amount

Output Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Roll amount

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Number of Rows to Roll Scroller (QsnRtvSclNumRoll) API returns the default number of rows to roll the scroller by for the Roll Scroller Up (QsnRollSclUp) and Roll Scroller Down (QsnRollSclDown) APIs. The default number of rows is specified on the session description. See "Create a Session (QsnCrtSsn) API" on page 227 and "Change Session (QsnChgSsn) API" on page 223 for details.

Authorities and Locks

None

Required Parameter

Session handle
INPUT; BINARY(4)

A handle for the session to be queried.

Omissible Parameter Group

Roll amount
OUTPUT; BINARY(4)

The variable that contains the number of scroller rows to roll by when the QsnRtvSclNumRoll API has completed.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Roll amount
OUTPUT; BINARY(4)

This API returns the value for the roll amount parameter if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Retrieve Session Data (QsnRtvSsnDta) API

Required Parameter:	
1	Session handle
Input	Binary(4)
Omissible Parameter Group:	
2	User data pointer
Output	PTR(SPP)
3	Error code
I/O	Char(*)
Returned Value:	
	User data pointer
Output	PTR(SPP)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Retrieve Session Data (QsnRtvSsnDta) API returns a pointer to the user data for the given session. The user data is the pointer specified on the session description and consists of user-specified data that is associated with the session. See "Format of the Session Description" on page 229 for details.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session for which the user data should be returned.

Omissible Parameter Group

User data pointer

OUTPUT; PTR(SPP)

A pointer to the user data, as specified on the session description, for the given session.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

User data pointer

OUTPUT; PTR(SPP)

This API returns the value for the user data pointer parameter if the operation was successful, or the null pointer if an error occurs.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1F E	Pointer is not on a 16 byte boundary.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA3A4 E	Specified window is not active.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Session Description (QsnRtvSsnD) API

Required Parameter Group:

1 Session handle

Input Binary(4)

2 Session description

Output Char(*)

3 Length of session description

Input Binary(4)

Omissible Parameter:

4 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Session Description (QsnRtvSsnD) API retrieves a copy of the session description for the given session. The session description may be different from the session description used when the Create a Session (QsnCrtSsn) or the Change Session (QsnChgSsn) API is called. The following fields will have actual values replacing 0 (if used):

Number of rows in scroller

Number of columns in scroller

Default number of rows to roll scroller by

Default number of columns to shift scroller by

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session for which the session description should be returned.

Session description
OUTPUT; CHAR(*)

The format of the data is shown in "Format of the Session Description Returned."

Length of session description
INPUT; BINARY(4)

The length of the session description parameter.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Format of the Session Description Returned

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(8)	Reserved

Offset		Type	Field
Dec	Hex		
16	10	CHAR(*)	Session description. The format of the remaining data returned is shown in “Format of the Session Description” on page 229.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | “Dynamic Screen Manager APIs,” on page 1 | APIs by category

Roll Scroller Down (QsnRollSclDown) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Roll amount

Input Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Roll Scroller Down (QsnRollSclDown) API rolls the scroller down by the specified number of scroller rows. A scroller row is distinct from a scroller line in that a scroller line consists of multiple scroller rows if line wrapping is set on and the line exceeds the width of the scroller.

Authorities and Locks

None

Required Parameter

Session handle
INPUT; BINARY(4)

A handle for the session to be rolled.

Omissible Parameter Group

Roll amount
INPUT; BINARY(4)

The number of scroller rows to roll the scroller by. If this parameter is omitted or set to 0, the default value is used. The default value can be queried using the Retrieve Number of Rows to Roll Scroller (QsnRtvSclNumRoll) API. If the roll amount would cause the scroller to roll past its top, then the top of the scroller will be displayed.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D3 E	Scroller not printed.
CPFA3D6 E	Session handle is incorrect.
CPFA3D8 E	Scroller display is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Roll Scroller Up (QsnRollSclUp) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Roll amount

Input Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Roll Scroller Up (QsnRollSclUp) API rolls the scroller up by the specified number of scroller rows. A scroller row is distinct from a scroller line in that a scroller line consists of multiple scroller rows if line wrapping is set on and the line exceeds the width of the scroller.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be rolled.

Omissible Parameter Group

Roll amount

INPUT; BINARY(4)

The number of scroller rows to roll the scroller by. If this parameter is omitted or set to 0, the default value is used. The default value can be queried using the Retrieve Number of Rows to Roll Scroller (QsnRtvSclNumRoll) API. If the roll amount causes the scroller to roll past its bottom, then the bottom of the scroller is displayed.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D3 E	Scroller not printed.
CPFA3D6 E	Session handle is incorrect.
CPFA3D8 E	Scroller display is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Shift Scroller left (QsnShfSclL) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Shift amount

Input Binary(4)

3 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Shift Scroller left (QsnShfSclL) API shifts the scroller to the left by the specified number of scroller columns. If line wrap mode is on, shifting has no effect.

Restrictions

If the low-level environment description (see “Format of the Low-Level Environment Description” on page 10) for the session specifies DBCS support or the device supports CCSID-based I/O, the shift amount parameter is ignored and the default value is used.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be shifted.

Omissible Parameter Group

Shift amount

INPUT; BINARY(4)

The number of scroller columns to shift the scroller by. If this parameter is omitted or set to 0, the default value is used. The default value can be queried using the Retrieve Number of Columns to Shift Scroller (QsnRtvSclNumShf) API. The scroller is shifted by the minimum of the shift amount and the number of scroller columns between the visible left column and the first column in the scroller.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.
CPFA3D8 E	Scroller display is not valid.

API introduced: V2R3

Top | “Dynamic Screen Manager APIs,” on page 1 | APIs by category

Shift Scroller Right (QsnShfSclR) API

Required Parameter:	
1	Session handle
Input	Binary(4)
Omissible Parameter Group:	
2	Shift amount
Input	Binary(4)
3	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Service Program: QSNAPI	
Default Public Authority: *USE	
Threadsafe: No	

The Shift Scroller Right (QsnShfSclR) API shifts the scroller to the right by the specified number of scroller columns. Any truncated data will become visible. If line wrap mode is on, shifting has no effect.

Restrictions

If the low-level environment description (see "Format of the Low-Level Environment Description" on page 10) for the session specifies DBCS support or the device supports CCSID-based I/O, the shift amount parameter is ignored and the default value is used.

Authorities and Locks

None.

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be shifted.

Omissible Parameter Group

Shift amount

INPUT; BINARY(4)

The number of scroller columns to shift the scroller by. If this parameter is omitted or set to 0, the default value is used. The default value can be queried using the Retrieve Number of Columns to Shift Scroller (QsnRtvSclNumShf) API. The scroller is shifted by the minimum of the shift amount

and the number of scroller columns between the visible center column and the last column (determined by the longest line currently visible) in the scroller.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.
CPFA3D8 E	Scroller display is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Toggle Line Wrap/Truncate Mode (QsnTglSclWrp) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter Group:

2 Wrap indication

Output Char(1)

3 Error code

I/O Char(*)

Returned Value:

Wrap indication

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Toggle Line Wrap/Truncate Mode (QsnTglSclWrp) API toggles the session between line wrap and truncation mode.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session to be queried.

Omissible Parameter Group

Wrap indication

OUTPUT; CHAR(1)

Indicates whether line wrap mode is on or off when the QsnTglSclWrp API has completed. The possible values are:

0	Line wrap mode is now off. Lines are truncated.
1	Line wrap mode is now on.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Wrap indication
OUTPUT; BINARY(4)

This API returns the value for the wrap indication parameter if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA340 E	Operation not supported with double-byte data.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Session I/O APIs

For additional information, see:

- Performance considerations
- Example: Create Session and Read Data

The session I/O APIs are:

- "Backspace on Scroller Line (QsnSclBS) API" on page 259 (QsnSclBS) sets the active position to the previous position in the current scroller line.
- "Go to Next Tab Position in Scroller Line (QsnSclTab) API" on page 260 (QsnSclTab) sets the active position to the next horizontal tab position.
- "Go to Start of Current Scroller Line (QsnSclCR) API" on page 262 (QsnSclCR) sets the active position to the start of the current scroller line.
- "Go to Start of Next Scroller Line (QsnSclNL) API" on page 263 (QsnSclNL) sets the active position to the start of the next scroller line.
- "Print Scroller Data (QsnPrtScl) API" on page 264 (QsnPrtScl) prints the scroller data.
- "Read Data from Session (QsnReadSsnDta) API" on page 266 (QsnReadSsnDta) reads the data from a session.
- » "Read Data from Session with CCSID (QsnReadSsnDtaCC) API" on page 268 (QsnReadSsnDtaCC) reads data in a particular CCSID from a session.«
- "Retrieve Session Line to Input Line (QsnRtvSsnLin) API" on page 270 (QsnRtvSsnLin) retrieves the input line from the scroller.
- "Start New Scroller Line at Current Position (QsnSclLF) API" on page 272 (QsnSclLF) sets the active position to the current position on the next scroller line.
- "Start New Scroller Page (QsnSclFF) API" on page 273 (QsnSclFF) starts a new scroller page.
- "Write Characters to Scroller (QsnWrtSclChr) API" on page 275 (QsnWrtSclChr) writes characters to the scroller.
- » "Write Characters to Scroller with CCSID (QsnWrtSclChrCC) API" on page 277 (QsnWrtSclChrCC) writes CCSID-encoded characters to the scroller.«

- "Write Line to Scroller (QsnWrtSclLin) API" on page 279 (QsnWrtSclLin) writes a data line to the scroller.
- » "Write Line to Scroller with CCSID (QsnWrtSclLinCC) API" on page 281 (QsnWrtSclLinCC) writes a CCSID-encoded data line to the scroller. «

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Backspace on Scroller Line (QsnSclBS) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Backspace on Scroller Line (QsnSclBS) API sets the active position to the previous position on the current scroller line. If the active position is at the start of the line, this operation has no effect.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session that the operation applies to.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Go to Next Tab Position in Scroller Line (QsnSclTab) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Go to Next Tab Position in Scroller Line (QsnSclTab) API sets the active position to the next horizontal tab position. Each tab interval is eight positions beyond the previous one, starting at the leftmost column in the scroller.

Authorities and Locks

None

Required Parameter

Session handle
INPUT; BINARY(4)

A handle for the session that the operation applies to.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3D6 E	Session handle is incorrect.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Go to Start of Current Scroller Line (QsnSclCR) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Go to Start of Current Scroller Line (QsnSclCR) API sets the active position to the start of the current scroller line.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session that the operation applies to.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Go to Start of Next Scroller Line (QsnSclNL) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Go to Start of Next Scroller Line (QsnSclNL) API sets the active position to the start of the next scroller line.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session that the operation applies to.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Print Scroller Data (QsnPrtScl) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Print Scroller Data (QsnPrtScl) API prints the entire contents of the scroller data to the default printer file. No printer output is produced if the scroller is empty. » CCSID-based data in the scroller is converted to the job CCSID before being printed. «

Restrictions

If the low-level environment description (see "Format of the Low-Level Environment Description" on page 10) for the session specifies DBCS support, or the device supports CCSID-based I/O, the printer file must have a record length greater than or equal to 4. A CPCA3D2 will be placed on the session message line otherwise. This can occur if the application uses the OVRPRTF (Override with Printer File) CL command. «

Authorities and Locks

None

Required Parameter

Session handle
INPUT; BINARY(4)

A handle for the session that the operation applies to.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA3D3 E	Scroller not printed.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Data from Session (QsnReadSsnDta) API

Required Parameter Group:

1	Session handle
Input	Binary(4)
2	Input buffer handle
Input	Binary(4)

Omissible Parameter Group:

3	Number of bytes read
Output	Binary(4)
4	Error code
I/O	Char(*)

Returned Value:

	Number of bytes read
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Data from Session (QsnReadSsnDta) API is used to read data from a session. A QsnReadInp operation is implicitly performed to read any field data. If the session has a DSM-defined input line, an implicit Clear Field Table (QsnClrFldTbl) operation is issued prior to redefining the session input line on each input operation. The data returned consists of only the data entered. That is, only the data from the cursor position within the field up to the last nonblank input character when an AID generating key is pressed is returned. If the session does not have a DSM- defined input line, data is read from any input fields defined on the screen, and all data, including blanks, is returned. In other respects, the processing of these user-defined input fields will be equivalent with the processing of the DSM-defined input line.

» If the previous input operation was a call to QsnReadSsnDtaCC, the session input line will be changed to support the data read by this API. «

If an AID key is pressed for which a corresponding function has been defined, this function will be called. Depending upon the return action specified, control would then return to the caller or another input operation will occur. See “Command Key Action Routines” on page 302 for details.

Restrictions

» If the device supports CCSID-based I/O and the session was not defined with an input line and the application defines input fields on the screen, DSM is not able to determine if the fields are CCSID-capable or which CCSID they support. Since CCSID-based data requires special datastream commands, DSM could send invalid data to the device, or otherwise corrupt or misinterpret the input data. Because of this, the data read from any application defined input fields on the screen will not be echoed to the scroller. Also, the default command key exit routines will not use this data. «

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session from which to read input. The session being read from must be the current window. You can use the Set Current Window (QsnSetCurWin) API to change the current window.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer to receive the result of the input operations if a direct operation is specified. The input buffer must have been created with the Create Input Buffer (QsnCrtInpBuf) API. The format of the data returned is the same as that of the Read Input Fields (QsnReadInp) API.

Omissible Parameter Group

Number of bytes read
OUTPUT; BINARY(4)

The number of bytes of data read. On a successful read operation, this value is the same as that returned by the Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API if passed the input buffer resulting from this operation.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Number of bytes read
OUTPUT; BINARY(4)

This API returns the value for the number of bytes read parameter if the operation was successful, -1 if there was a general failure, or -2 if the invite active flag is on in the associated environment and the read from invited device operation timed out.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3D6 E	Session handle is incorrect.
CPFA3D9 E	Error calling the command key action routine.

For examples of Read Data from Session APIs, see Example: Create Session and Read Data.

API Introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Read Data from Session with CCSID (QsnReadSsnDtaCC) API

Required Parameter Group:

1	Session handle
Input	Binary(4)
2	Input buffer handle
Input	Binary(4)

Omissible Parameter Group:

3	CCSID
Input	Binary(4)
4	Number of bytes read
Output	Binary(4)
5	Error code
I/O	Char(*)

Returned Value:

Number of bytes read

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Read Data from Session with CCSID (QsnReadSsnDtaCC) API is used to read data from a session. A QsnReadInp operation is implicitly performed to read any field data. If the session has a DSM-defined input line, an implicit Clear Field Table (QsnClrFldTbl) operation is issued prior to redefining the session input line on each input operation. The data returned consists of only the data entered. That is, only the data from the cursor position within the field up to the last nonblank input character when an AID generating key is pressed is returned. If the session does not have a DSM-defined input line, data is read from any input fields defined on the screen, and all data, including blanks, is returned. In other respects, the processing of these user-defined input fields will be equivalent with the processing of the DSM-defined input line.

If the previous input operation was a call to QsnReadSsnDta, the session input line will be changed to support the CCSID requested.

If an AID key is pressed for which a corresponding function has been defined, this function will be called. Depending upon the return action specified, control would then return to the caller or another input operation will occur. See "Command Key Action Routines" on page 302 for details.

Restrictions

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

The CCSID value given must be supported by the device or emulator, otherwise a CPF3BDE will be signaled.

If the device supports CCSID-based I/O and the session was not defined with an input line and the application defines input fields on the screen, DSM is not able to determine if the fields are CCSID-capable or what the CCSID is. Since CCSID-based data requires special datastream commands, DSM could send invalid data to the device, or otherwise corrupt or misinterpret the input data. Because of this, the data read from any application defined input fields on the screen will not be echoed to the scroller. Also, the default command key exit routines will not use this data.

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session from which to read input. The session being read from must be the current window. You can use the Set Current Window (QsnSetCurWin) API to change the current window.

Input buffer handle
INPUT; BINARY(4)

A handle for the input buffer to receive the result of the input operations if a direct operation is specified. The input buffer must have been created with the Create Input Buffer (QsnCrtInpBuf) API. The format of the data returned is the same as that of the Read Input Fields (QsnReadInp) API.

Omissible Parameter Group

CCSID
INPUT; BINARY(4)

The CCSID of the data to read. The session input line will be changed to support a CCSID-based read, unless a previous QsnReadSsnDtaCC was performed for this CCSID.

If the CCSID given is not supported by the device, a CPF3BDE is signaled.

If this parameter is omitted (zero is passed in as the CCSID), the job CCSID is used. If the job CCSID is 65535, the default job CCSID is used instead.

Number of bytes read
OUTPUT; BINARY(4)

The number of bytes of data read. On a successful read operation, this value is the same as that returned by the Retrieve Length of Field Data in Buffer (QsnRtvFldDtaLen) API if passed the input buffer resulting from this operation.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Number of bytes read
OUTPUT; BINARY(4)

This API returns the value for the number of bytes read parameter if the operation was successful, -1 if there was a general failure, or -2 if the invite active flag is on in the associated environment and the read from invited device operation timed out.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3BDE E	CCSID &1 not supported by API.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3A4 E	Specified window is not active.
CPFA3D6 E	Session handle is incorrect.
CPFA3D9 E	Error calling the command key action routine.

For examples of Read Data from Session APIs, see Create Session and Read Data—Example.

API Introduced: V5R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Retrieve Session Line to Input Line (QsnRtvSsnLin) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Retrieve Session Line to Input Line (QsnRtvSsnLin) API retrieves the input line from the scroller that corresponds to the cursor position within the scroller. If the cursor is outside the scroller and the retrieve request directly follows another retrieve with no intervening I/O operations, then the line before the line previously retrieved is returned. Otherwise, the last input line is retrieved. If there is no input data, this API still completes successfully.

Restrictions

The data on the retrieved line may be incompatible with the current state of the session input line. This may happen when QsnReadSsnDta echoes input to the scroller, and that data is later retreived on a call or after a call of QsnReadSsnDtaCC. If the session input line does not support the data of the retrieved line, CPFA3E1 will be signaled and placed on the session message line. The scroller line will not be retrieved to the session input line, since the data can be corrupted or misinterpreted. «

Authorities and Locks

None

Required Parameter

Session handle
INPUT; BINARY(4)

A handle for the session for which to retrieve the input line.

Omissible Parameter

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA3A4 E	Specified window is not active.
CPFA3D6 E	Session handle is incorrect.
» CPFA3E1 E	Data not compatible with session input line. «

API Introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Start New Scroller Line at Current Position (QsnSclLF) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Start New Scroller Line at Current Position (QsnSclLF) API sets the active position to the current position on the next scroller line.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session that the operation applies to.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Start New Scroller Page (QsnSclFF) API

Required Parameter:

1 Session handle

Input Binary(4)

Omissible Parameter:

2 Error code

I/O Char(*)

Returned Value:

Return code

Output Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Start New Scroller Page (QsnSclFF) API starts a new scroller page. Any data currently on the session is rolled off the top of the scroller, but can still be viewed by rolling the scroller up.

Authorities and Locks

None

Required Parameter

Session handle

INPUT; BINARY(4)

A handle for the session for which to start the new page.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Characters to Scroller (QsnWrtSclChr) API

Required Parameter Group:	
1	Session handle
Input	Binary(4)
2	Data
Input	Char(*)
3	Data length
Input	Binary(4)
Omissible Parameter:	
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Write Characters to Scroller (QsnWrtSclChr) API writes one or more characters to the scroller starting at the active position. The active position following this operation is one position past the last character written or that specified by a control character sequence if it appears at the end of the data. If the entire data string cannot fit in the scroller buffer, no portion of the string will be written.

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session to which the scroller characters are to be written.

Data Input; CHAR(*)

The characters to be written to the scroller. If the data does not fit within the width of the session window, it is wrapped across multiple lines or truncated, depending on the value of the wrap indication field on the session description.

Data length
Input; CHAR(*)

The length of the data parameter.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA342 E	No matching shift-in or shift-out character found.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.
CPFA3D7 E	Data for scroller is too long.
CPG3264 D	DBCS character string does not have even length.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Characters to Scroller with CCSID (QsnWrtSclChrCC) API

Required Parameter Group:	
1	Session handle
Input	Binary(4)
2	Data
Input	Char(*)
3	Data length
Input	Binary(4)
Omissible Parameter:	
4	CCSID
Input	Binary(4)
5	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Write Characters to Scroller with CCSID (QsnWrtSclChrCC) API writes one or more characters to the scroller starting at the active position. The active position following this operation is one position past the last character written or that specified by a control character sequence if it appears at the end of the data. If the entire data string cannot fit in the scroller buffer, no portion of the string will be written.

Note: CDRA conversion is not performed upon this data.

Restrictions

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

The CCSID value given must be supported by the device or emulator, otherwise a CPF3BDE will be signaled.

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session to which the scroller characters are to be written.

Data Input; CHAR(*)

The characters in the CCSID given by the CCSID parameter to be written to the scroller. If the data does not fit within the width of the session window, it is wrapped across multiple lines or truncated, depending on the value of the wrap indication field on the session description.

Data length
Input; CHAR(*)

The length of the data parameter.

Omissible Parameter

CCSID
INPUT; BINARY(4)

The CCSID of the data to be written. If the CCSID given is not supported by the device, a CPF3BDE is signaled.

If this parameter is omitted (zero is passed in as the CCSID), the job CCSID is used. If the job CCSID is 65535, the default job CCSID is used instead.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.
Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3BDE E	CCSID &1 not supported by API.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.
CPFA3D7 E	Data for scroller is too long.

API Introduced: V5R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Line to Scroller (QsnWrtSclLin) API

Required Parameter Group:	
1	Session handle
Input	Binary(4)
2	Line data
Input	Char(*)
3	Line data length
Input	Binary(4)
Omissible Parameter:	
4	Error code
I/O	Char(*)
Returned Value:	
	Return code
Output	Binary(4)
Default Public Authority: *USE	
Service Program: QSNAPI	
Threadsafe: No	

The Write Line to Scroller (QsnWrtSclLin) API writes a line of data, such as an informational message, to the scroller. The data is written starting at the first position on the next scroller line. The active position after this operation is the start of the next scroller line following the row containing the last data character written, or specified by a control character sequence if one appears at the end of the data. If the entire line cannot fit in the scroller buffer, no portion of the data will be written.

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session to which the scroller line is to be written.

Line data
Input; CHAR(*)

The data to be written to the scroller. If the line does not fit within the width of the session window, it is wrapped across multiple lines or truncated, depending on the value of the wrap indication field on the session description.

Note: The first 2 bytes of the scroller are reserved for the prefix area to the left of the scroller line.

Line data length
Input; CHAR(*)

The length of the line data parameter.

Note: The first 2 bytes of the scroller are reserved for the prefix area to the left of the scroller line.

Omissible Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Returned Value

Return code

OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA342 E	No matching shift-in or shift-out character found.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.
CPFA3D7 E	Data for scroller is too long.
CPG3264 D	DBCS character string does not have even length.

API introduced: V2R3

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Write Line to Scroller with CCSID (QsnWrtSclLinCC) API

Required Parameter Group:

1	Session handle
Input	Binary(4)
2	Line data
Input	Char(*)
3	Line data length
Input	Binary(4)

Omissible Parameter:

4	CCSID
Input	Binary(4)
5	Error code
I/O	Char(*)

Returned Value:

	Return code
Output	Binary(4)

Default Public Authority: *USE

Service Program: QSNAPI

Threadsafe: No

The Write Line to Scroller with CCSID (QsnWrtSclLinCC) API writes a line of data, such as an informational message, to the scroller. The data is written starting at the first position on the next scroller line. The active position after this operation is the start of the next scroller line following the row containing the last data character written, or specified by a control character sequence if one appears at the end of the data. If the entire line cannot fit in the scroller buffer, no portion of the data will be written.

Note: CDRA conversion is not performed upon this data.

Restrictions

This command is not supported by all control units. A CPFA306 error occurs if an attempt is made to issue this command to a control unit that does not support it.

The CCSID value given must be supported by the device or emulator, otherwise a CPF3BDE will be signaled.

Authorities and Locks

None

Required Parameter Group

Session handle
INPUT; BINARY(4)

A handle for the session to which the scroller line is to be written.

Line data
Input; CHAR(*)

The data in the CCSID given by the CCSID parameter to be written to the scroller. If the line does not fit within the width of the session window, it is wrapped across multiple lines or truncated, depending on the value of the wrap indication field on the session description.

Note: The first 2 bytes of the scroller are reserved for the prefix area to the left of the scroller line.

Line data length
Input; CHAR(*)

The length of the line data parameter.

Note: The first 2 bytes of the scroller are reserved for the prefix area to the left of the scroller line.

Omissible Parameter

CCSID
INPUT; BINARY(4)

The CCSID of the data to be written. If the CCSID given is not supported by the device, a CPF3BDE is signaled.

If this parameter is omitted (zero is passed in as the CCSID), the job CCSID is used. If the job CCSID is 65535, the default job CCSID is used instead.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.
Returned Value

Return code
OUTPUT; BINARY(4)

A return code indicating the result of the operation. The value returned will be 0 if the operation was successful, or -1 otherwise.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3BDE E	CCSID &1 not supported by API.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFA333 E	Parameter &1 not positive integer value.
CPFA31E E	Required parameter &1 omitted.
CPFA343 E	Output operation not done.
CPFA344 E	The file &2 in library &3 is not valid.
CPFA345 E	The invite active flag is not valid.
CPFA3D6 E	Session handle is incorrect.
CPFA3D7 E	Data for scroller is too long.

API Introduced: V5R3

Concepts

These are the concepts for this category.

Using Dynamic Screen Manager APIs

Data Structures for DSM APIs

Data structures for use with ILE C, ILE COBOL, and ILE RPG/400(R) are available in the QSYSINC library in member QSNAPI for service program QSNAPI.

Omitting Parameters with Associated Lengths

To omit a parameter with an associated length parameter, that length parameter should be omitted or specified as 0. If the length parameter is specified with a value greater than 0, the parameter with which it is associated is required.

For example, to omit the user extension information on the low-level environment, specify either a NULL pointer by value, or 0 by reference for the length. The extension information structure is ignored. If the length is greater than 0, the extension information structure cannot be NULL. If it is, then a `Required parameter omitted` error is generated.

"Dynamic Screen Manager APIs," on page 1 | APIs by category

Low-Level Services Examples

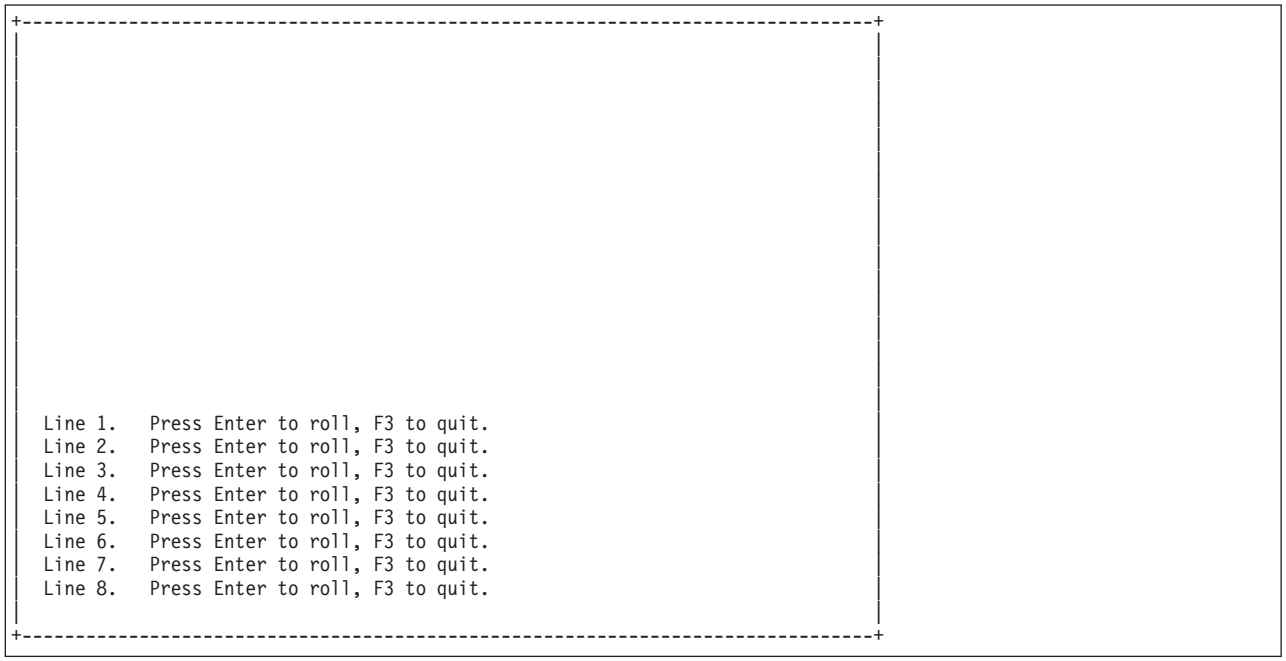
Low-Level Services Example—1

The sample ILE C program in Figure 1 shows how to use the Write Data (QsnWrtDta), Get AID (QsnGetAID), and Roll Up (QsnRollUp) APIs. The program writes a line at the bottom of the screen and if F3 is not pressed, rolls the screen up and writes a new line at the bottom of the screen. The roll area for the QsnRollUp API is defined to be from row 1 to 24 and one line is rolled up. If F3 is pressed, the program ends. A partial screen display is shown in Figure 2 (page 284).

Figure 1. Program to Roll Text on Screen

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "qsnapi.h"

int main(void)
{
    long i;
    char s[100];

    QsnClrScr('0', 0, 0, NULL);
    for (i = 1; ; ++i) {
        sprintf(s, "Line %2.d.   Press Enter to roll, F3 to quit.", i);
        QsnWrtDta(s, strlen(s), 0, 24, 2, QSN_SA_NORM, QSN_SA_NORM,
                  QSN_SA_NORM, QSN_SA_NORM, 0, 0, NULL);
        if (QsnGetAID(NULL, 0, NULL) == QSN_F3)
            break;
        QsnRollUp(1, 1, 24, 0, 0, NULL);
    }
}
```

Figure 2. Screen Output from Roll Text Program



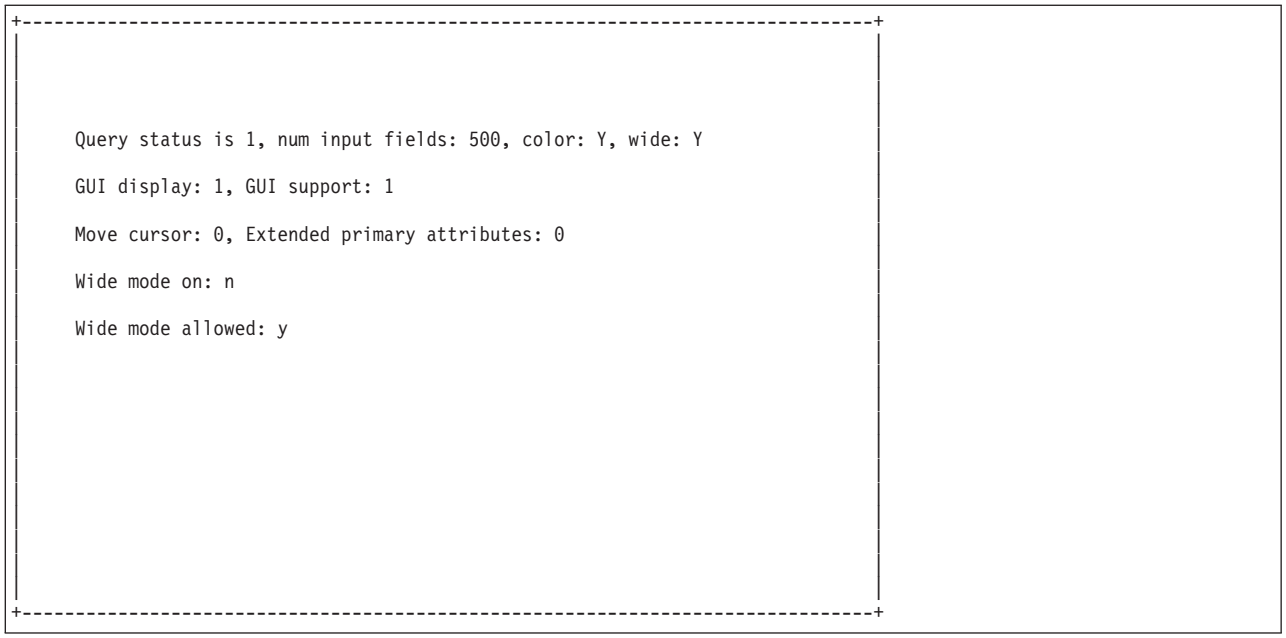
Low-Level Services Example—2

Figure 3 shows how to use the Query 5250 (QsnQry5250) API. A sample display is shown in Figure 4 (page 285).

Figure 3. Program Using the Query 5250 (QsnQry5250) API

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "qsnapi.h"

#define    TRUE    1
#define    FALSE   0

int main(void)
{
    Qsn_Qry_5250_T qry5250;
    Qsn_WSC_display_T *dsp = (Qsn_WSC_display_T *)(qry5250.WSC_display);
    char s[100];
    Q_Bin4 row;
    Qsn_Cmd_Buf_T buf;

    QsnQry5250(&qry5250, sizeof(qry5250), NULL);
    buf = QsnCrtCmdBuf(100, 20, 0, NULL, NULL);
    QsnClrScr('0', buf, Q_NO_HANDLE, NULL);
    sprintf(s, "Query status is %c, num input fields: %d, color: %c, wide: %c",
            qry5250.query_status, qry5250.num_input_capable,
            (QsnQryColorSup(NULL, Q_NO_HANDLE, NULL) == TRUE ? 'Y' : 'N'),
            (dsp->scr_size == 3 ? 'Y' : 'N'));
    QsnWrtDta(s, strlen(s), 0, row=5, 5, QSN_SA_NORM, QSN_SA_NORM,
                QSN_SA_NORM, QSN_SA_NORM, buf, Q_NO_HANDLE, NULL);
    sprintf(s, "GUI display: %d, GUI support: %d",
               dsp->GUI_display, dsp->GUI_support);
    QsnWrtDta(s, strlen(s), 0, row+=2, 5, QSN_SA_NORM, QSN_SA_NORM,
                QSN_SA_NORM, QSN_SA_NORM, buf, Q_NO_HANDLE, NULL);
    sprintf(s, "Move cursor: %d, Row1/Col1: %d, ReadMDTImmAlt: %d",
               dsp->move_csr_order, dsp->row1_col1, dsp->Read_MDT_Imm_Alt);
    QsnWrtDta(s, strlen(s), 0, row +=2, 5, QSN_SA_NORM, QSN_SA_NORM,
                QSN_SA_NORM, QSN_SA_NORM, buf, Q_NO_HANDLE, NULL);
    sprintf(s, "Extended primary attributes: %d, DBCS: %x",
               dsp->extended_pri_atr_DP, dsp->DBCS);
    QsnWrtDta(s, strlen(s), 0, row +=2, 5, QSN_SA_NORM, QSN_SA_NORM,
                QSN_SA_NORM, QSN_SA_NORM, buf, Q_NO_HANDLE, NULL);
    sprintf(s, "Wide mode on: %c",
               QsnRtvMod(NULL, Q_NO_HANDLE, NULL) == QSN_DSP04 ? 'y' : 'n');
    QsnWrtDta(s, strlen(s), 0, row +=2, 5, QSN_SA_NORM, QSN_SA_NORM,
                QSN_SA_NORM, QSN_SA_NORM, buf, Q_NO_HANDLE, NULL);
    sprintf(s, "Wide mode allowed: %c",
               QsnQryModSup(QSN_DSP04, NULL, Q_NO_HANDLE, NULL) ==
                                                  TRUE ? 'y' : 'n');
    QsnWrtDta(s, strlen(s), 0, 13, 5, QSN_SA_NORM, QSN_SA_NORM,
                QSN_SA_NORM, QSN_SA_NORM, buf, Q_NO_HANDLE, NULL);
    QsnPutBuf(buf, Q_NO_HANDLE, NULL);
    QsnGetAID(NULL, Q_NO_HANDLE, NULL);
}
```

Figure 4. Screen Output from QsnQry5250 API Program



Low-Level Services Example—3

The sample program in Figure 5 shows how to use the Read Modified Fields (QsnReadMDT) API in conjunction with the buffer query APIs. The resulting screen display before and after the input operations is shown in Figure 6 (page 288) and Figure 7 (page 288), respectively.

Figure 5. Program Using Read Modified Fields (QsnReadMDT) API

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "qsnapi.h"

#define    TRUE     1
#define    FALSE    0

int main(void)
{
    Q_Bin4 i, t1, t2, t3, numflds;
    const Q_Uchar cc1=QSN_CC1_MDTALL_CLRALL, cc2=QSN_CC2_UNLOCKBD;
    const Q_Uchar nosa = QSN_NO_SA, norm = QSN_SA_NORM, saul = QSN_SA_UL;
    char pad = ' ';
    Qsn_Cmd_Buf_T cmdbuf, cmdbuf2;
    Qsn_Inp_Buf_T inpbuf;
    Qsn_Fld_Inf_T fldqry;
    Qsn_Read_Inf_T rdqry;
    char msg[100];
    Q_Fdbk_T fdbk = { sizeof(Q_Fdbk_T) };

    cmdbuf = QsnCrtCmdBuf(100, 50, 0, NULL, NULL);
    cmdbuf2 = QsnCrtCmdBuf(100, 50, 0, NULL, NULL);
    inpbuf = QsnCrtInpBuf(200, 50, 0, NULL, NULL);
    QsnClrScr('0', 0, 0, NULL);
    QsnWTD(cc1, cc2, cmdbuf, 0, NULL);
    while (TRUE) {
        QsnSetFld(0, 10, 3, 2, QSN_FFW_ALPHA_SHIFT, NULL, 0, saul,
                  saul, cmdbuf, 0, NULL);
        QsnSetFld(0, 10, 5, 2, QSN_FFW_ALPHA_SHIFT, NULL, 0, saul,
                  saul, cmdbuf, 0, NULL);
        QsnSetFld(0, 10, 7, 2, QSN_FFW_ALPHA_SHIFT, NULL, 0, saul,
                  saul, cmdbuf, 0, NULL);
        numflds = QsnReadMDT(QSN_CC1_NULL, QSN_CC1_NULL, NULL,
                             inpbuf, cmdbuf, 0, NULL);
        if (QsnRtvReadAID(inpbuf, NULL, NULL) == QSN_F3)
            break;
        QsnPutBuf(cmdbuf2, 0, NULL);
        QsnClrBuf(cmdbuf2, NULL);
        QsnClrBuf(cmdbuf, NULL);
        QsnWTD(cc1, cc2, cmdbuf, 0, NULL);
        sprintf(msg, "Num Fields Change: %d", numflds);
        QsnWrtDta(msg, strlen(msg), 0, 2, 30, norm, norm, norm, norm,
                  cmdbuf, 0, NULL);
        for (i = 1; i <= numflds; i++) {
            fldqry.len = 0;
            if (QsnRtvFldInf(inpbuf, i, &fldqry, sizeof(fldqry),
                             0, &fdbk) != QSN_FAIL) {
                sprintf(msg,
                        "Field# %d, row %d, col %d, len %d, value %.*s",
                        i, fldqry.row, fldqry.col, fldqry.len,
                        fldqry.len, fldqry.data);
                QsnWrtDta(msg, t1=strlen(msg), 0,
                     t2=i+3, t3=30, norm, norm, norm, norm,
                     cmdbuf, 0, NULL);
                QsnWrtPad(pad, t1, 0, t2, t3, cmdbuf2, 0, NULL);
            } else {
                sprintf(msg, "Field query failed");
                QsnWrtDta(msg, t1=strlen(msg), 0,
                     t2=4, t3=30, norm, norm, norm, norm,
                     cmdbuf, 0, NULL);
                QsnWrtPad(pad, t1, 0, t2, t3, cmdbuf2, 0, NULL);
            }
        }

        QsnRtvReadInf(inpbuf, &rdqry, sizeof(rdqry), 0, NULL);
        sprintf(msg, "Read information:");
        QsnWrtDta(msg, strlen(msg), 0, t2=10, t3=2, norm, norm,
                  norm, norm, cmdbuf, 0, NULL);
        QsnWrtPadAdr(pad, -1, -1, t2, t3, cmdbuf2, 0, NULL);
        sprintf(msg, "Bytes returned %d, available: %d",
                rdqry.bytes_returned, rdqry.bytes_available);
        QsnWrtDta(msg, strlen(msg), 0, ++t2, t3+2, norm, norm,
                norm, norm, cmdbuf, 0, NULL);
        sprintf(msg, "First data byte: %p", rdqry.dta);
        QsnWrtDta(msg, strlen(msg), 0, ++t2, t3+2, norm, norm,
                norm, norm, cmdbuf, 0, NULL);
        sprintf(msg, "First field byte: %p", rdqry.fld_dta);
        QsnWrtDta(msg, strlen(msg), 0, ++t2, t3+4, norm, norm,
                norm, norm, cmdbuf, 0, NULL);
        sprintf(msg, "Diff: %d", rdqry.fld_dta-rdqry.dta);
        QsnWrtDta(msg, strlen(msg), 0, ++t2, t3+4, norm, norm,
```

Figure 6. Display Screen before Input Operation

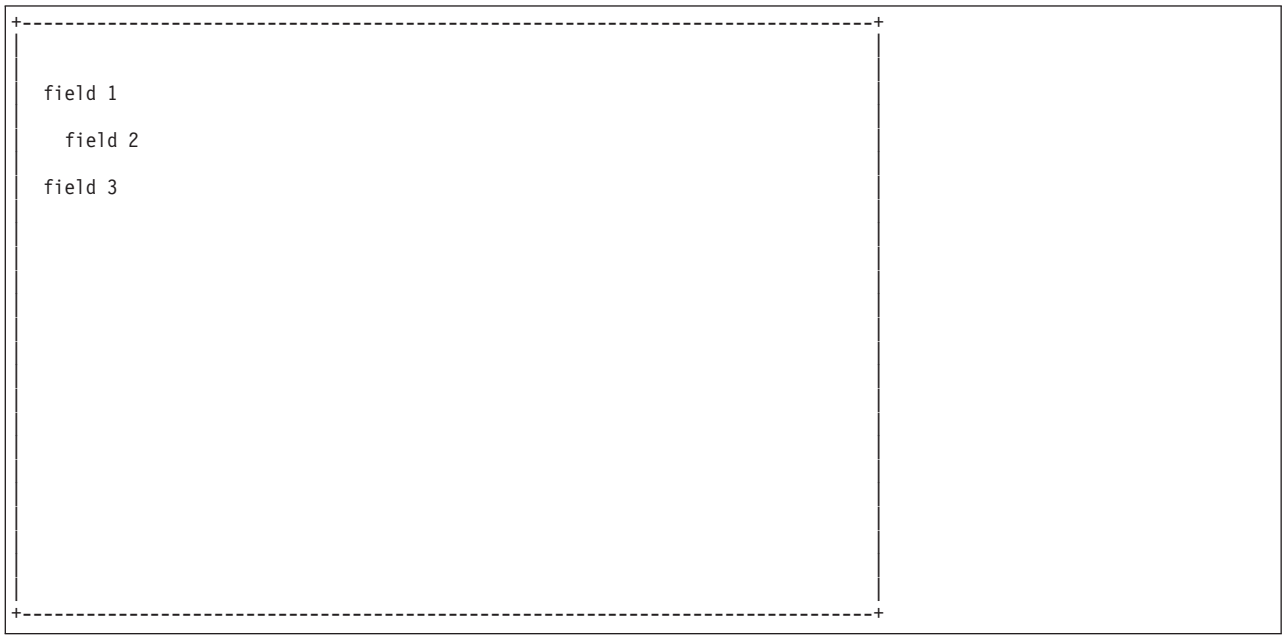
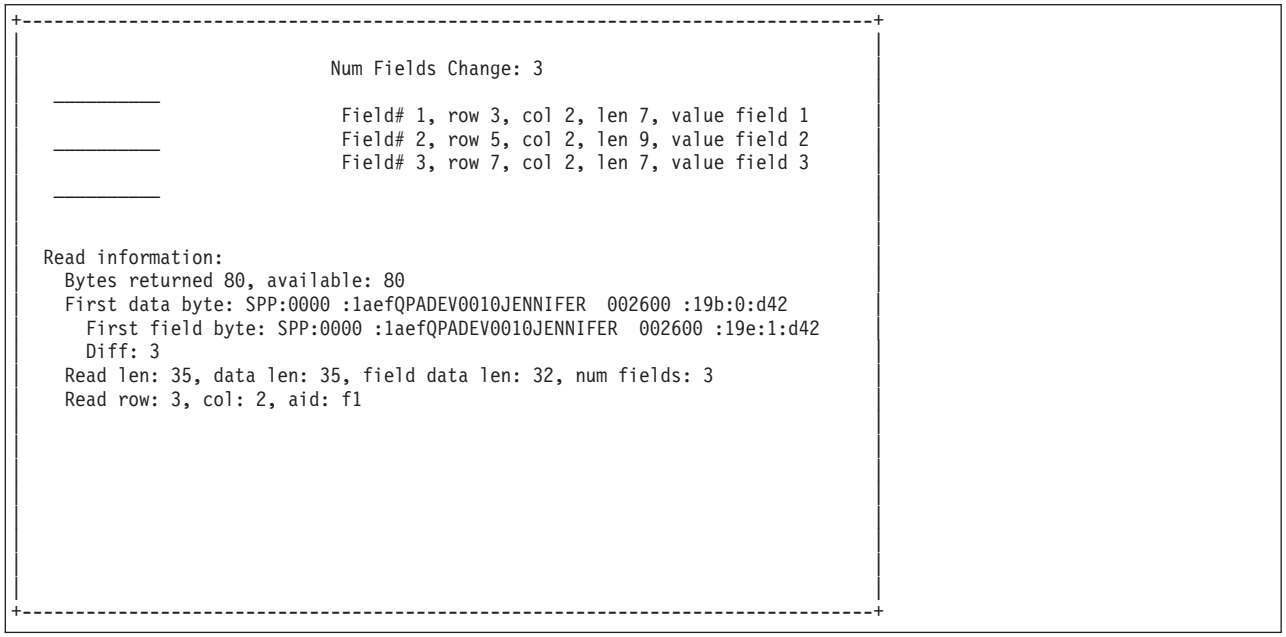


Figure 7. Display Screen after Input Operation



Low-Level Services Example—4

The sample program in Figure 8 shows how to use the QsnWrtSFMaj and QsnWrtSFMin APIs to display a window on the screen using the 5250 Create Window command. See the *5494 Remote Control Unit Functions Reference R3.1*, SC30-3533-04, for more information. This book can be viewed online through the IBM Publications Center .

Figure 8. Program Using Read QsnWrtSFMaj and QsnWrtSFMin APIs

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "qsnapi.h"

#define STRUCTURED_FIELD                '\xd9'
#define CREATE_WIN_STRUCTURED_FIELD     '\x51'
#define WINDOW_CURSOR_RESTRICT          '\x80'
#define WINDOW_PULL_DOWN                '\x40'
#define WINDOW_BORDER_MINOR             '\x01'
#define WINDOW_TITLE_MINOR              '\x10'

typedef _Packed struct {
    Q_Bin2 length;
    Q_Uchar class;
    Q_Uchar type;
    Q_Uchar flag1;
    Q_Uchar flag2;
    Q_Uchar reserved;
    Q_Uchar num_rows;
    Q_Uchar num_cols;
} create_window_major;

typedef _Packed struct {
    Q_Uchar length;
    Q_Uchar type;
    Q_Uchar flag1;
    Q_Uchar mono_attr;
    Q_Uchar color_attr;
    Q_Uchar ul;
    Q_Uchar top;
    Q_Uchar ur;
    Q_Uchar left;
    Q_Uchar right;
    Q_Uchar ll;
    Q_Uchar bottom;
    Q_Uchar lr;
} create_window_border_minor;

typedef _Packed struct {
    Q_Uchar length;
    Q_Uchar type;
    Q_Uchar flag1;
    Q_Uchar mono_attr;
    Q_Uchar color_attr;
    Q_Uchar reserved;
} create_window_title_minor;

int main(void)
{
    create_window_major win;
    create_window_border_minor win_border;
    create_window_title_minor win_title;
    Qsn_Cmd_Buf_T cmdbuf;
    Q_Uchar c1;
    char title_text[] = "Title";

    cmdbuf = QsnCrtCmdBuf( 300, 20, 0, NULL, NULL);

    /* setup Create window command major structure */
    win.length = sizeof(win);
    win.class = STRUCTURED_FIELD;
    win.type = CREATE_WIN_STRUCTURED_FIELD;
    win.flag1 = WINDOW_CURSOR_RESTRICT;
    win.flag2 = '\x00';
    win.reserved = '\x00';
    win.num_rows = 10;
    win.num_cols = 10;

    /* write Create window command major structure to command buffer */
    QsnWrtSFMaj((char *)(&win), sizeof(win), 0, 9, 22, cmdbuf, 0, NULL);

    /* setup border presentation minor structure */
    win_border.length = sizeof(win_border);
    win_border.type = WINDOW_BORDER_MINOR;
    win_border.flag1 = '\x00';
    win_border.mono_attr = QSN_SA_RI;
    win_border.color_attr = QSN_SA_PNK;
    win_border.ul = '+';
    win_border.top = '*';
    win_border.ur = '+';
    win_border.left = '-';
```

Using Low-Level Screen I/O Services APIs

DSM Error Handling

Calls to most of the interfaces can result in a direct I/O operation, or in the storing of commands in a command buffer. The command buffer provides a way of saving the commands so that multiple operations can be specified and performed in a single I/O operation. DSM performs error handling as much as possible prior to issuing an I/O operation. For example, if a request is made to place the screen in wide mode, and the display does not support this mode, DSM detects and reports the error condition before performing an I/O operation. This way of handling errors is particularly useful in the case where multiple commands have been saved in a buffer. Otherwise, there is no obvious way to determine which command was in error when the I/O operation fails.

The errors that can occur for each operation are listed with the operation. If an error message indicates that the error is issued for a negative response code, this means that the error was not detected by DSM and occurred on the I/O operation.

Note: When you are using the OS/400(R) TELNET display station emulation, an unsuccessful I/O operation may be undetected initially, but will be reported on the next operation.

Device Support

The 5250 Query command is used to determine the valid commands for a particular device. This command is issued for the current device at the start of each DSM session and the information is saved for subsequent queries. If the 5250 Query command is not supported, the base data stream support as documented in the 5250 Functions Reference is assumed, with color and wide support being determined by the device type.

Operating Environments

The low-level interfaces operate within an environment that can be defined using the Create Low-Level Environment (QsnCrtEnv) API. The low-level environment defines the operating modes, such as DBCS support and the window mode. The environment is passed as a parameter to most of the low-level services APIs. There is no need to define a low-level environment unless you need a specific operating environment that is different from the default. The default low-level environment is indicated on the low-level service APIs by specifying the environment handle as zero.

Direct and Indirect Operations

Many of the low-level APIs accept an optional command buffer as a parameter. For such APIs, the command buffer can be used to store and accumulate a group of requested operations. The accumulated operations can then be written to the screen in a single I/O operation. Better performance can be achieved because a group of repetitive operations can be issued to the screen without having to recall the sequence of individual APIs for each repeated operation.

A **direct operation** is one that omits the command buffer. The requested operation takes place immediately. For most APIs, specifying the command buffer results in an **indirect operation**. No I/O operation takes place and the operation is simply stored in the command buffer. Several of the screen input APIs, however, do perform a direct operation when the command buffer is specified. This semantic is discussed in "Read Input Fields (QsnReadInp) API" on page 100.

DBCS Considerations

You can write DBCS data enclosed with SO/SI to the screen, and when the underlying display supports it, graphic DBCS using the Write Data (QsnWrtDta) API to specify the data stream Write Extended Attribute order. (See the 5250 data stream documentation for further details.) You can define fields as being DBCS through the Set Field (QsnSetFld) API using the appropriate field control word. DBCS data can be written to such fields as described above. If you specify DBCS support on the low-level environment description, (see "Format of the Low-Level Environment Description" on page 10), the APIs will handle the parsing of DBCS data and fields appropriately.

The APIs do not provide any special processing of DBCS data, such as adding SO/SI to DBCS graphic data when graphic data is not supported by the display. For example, if you want to define a field as graphic DBCS and write graphic DBCS data to it, code the QsnSetFld API specifying a control word of QSN_FCS_DBCS_PURE (x'8220') and then write the graphic data to a command buffer using the QsnWrtDta API. Precede and follow this data with Write Extended Attribute orders to add the extended NLS SO/SI attributes. If you want to write a graphic data value to a non-graphic DBCS field, you must enclose the graphic DBCS data with SO/SI prior to calling the QsnWrtDta API.

Performance Considerations

The following operations can incur overhead and adversely affect the performance of your application:

- **QsnCrtEnv**

  Specifying translation of x'3f' to x'1f' can incur overhead because all outgoing data must be checked for this value. This option should be specified only if CDRA is on and translation between the code pages will result in a x'3f' occurrence in data to be displayed.
- **QsnSavScr**

  This operation results in the entire contents of the screen being read, which can adversely affect response time. This is typically about 3KB, but could be up to 28KB.
- **QsnRstScr**

  This operation writes the result of a save screen back to the device, which can adversely affect response time.

If you have GUI support, you can put additional commands after the QsnSavScr or QsnRstScr APIs to reduce I/O operations and improve performance.

Deleting structures associated with handles, such as command buffers, prior to exiting a program will improve performance for the programs that use APIs that create handles.

Limitations and Restrictions

The following limitations or restrictions apply to the low-level interfaces:

- Certain functions are supported by control units that do not support the 5250 Query command. If the Query command is not supported, it is assumed that the particular function is not supported either. These functions are transparent data support and move cursor order support. Device attributes such as wide mode and color support are determined based on the device type if the 5250 Query command is not supported.
- For the Retrieve Display Mode (QsnRtvMod) operation to correctly report the current state of the display, all commands that affect this state (such as a Clear Unit or Clear Unit Alternate) must occur as the first command in any command stream written to the display. This is because the work station control unit inspects the first command in the stream to determine if a state change is taking place. Most iSeries(TM) programs, including the DSM APIs, send these commands only at the beginning of a

stream. If you write a stream in which such commands do not appear at the beginning of the stream, the results of the Retrieve Display Mode (QsnRtvMod) operation may not be accurate.

- When conversions are performed, they are performed only after a Read Input Fields (QsnReadInp), Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), Read Immediate (QsnReadImm), or Read Modified Immediate Alternate (QsnReadMDTImmAlt) operation. They are performed on all incoming field data, including transparent and numeric data. You must turn conversion on and off. To prevent certain data from being converted, you explicitly set the conversion options on the QsnCrtEnv and QsnChgEnv APIs. The conversions that are affected by this are CDRA conversion based on the job CCSID and conversions of X'1F' in the incoming data stream to X'3F'.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

5250 Data Stream Details

AID-Generating Keys

The AID (attention indicator) code identifies to the host system the function being requested from the keyboard. The AID code is returned by certain input operations when the operator presses an AID-generating key. The following table lists the AID-generating keys and the AID codes associated with each key. See "Format of the Low-Level Environment Description" on page 10 for instructions on how to specify an alternative help key.

AID Codes		
AID key	**Mnemonic**	**AID Code**
Cmd 1 - 12 (cmd 1=x'31', cmd12=x'3C')	QSN_F1 - QSN_F12	x'31' - x'3C'
Selector Light Pen Auto Enter	QSN_SLP	x'3F'
Forward Edge Trigger Auto Enter	QSN_FET	x'50'
PA1	QSN_PA1	x'6C'
PA2	QSN_PA2	x'6E'
PA3	QSN_PA3	x'6B'
Cmd 13 - 24 (cmd 13=x'B1', cmd24=x'BC')	QSN_F13 - QSN_F24	x'B1' - x'BC'
Clear	QSN_CLEAR	x'BD'
Enter or Record Advance	QSN_ENTER	x'F1'
Help (not in error state)	QSN_HELP	x'F3'
Roll Down or Page Up	QSN_ROLLDOWN or QSN_PAGEUP	x'F4'
Roll Up or Page Down	QSN_ROLLUP or QSN_PAGEDOWN	x'F5'
Print	QSN_PRINT	x'F6'
Record Backspace	QSN_RECBS	x'F8'

Control Characters

The display control characters (CCs) are always specified as a pair of 1-byte fields. They are used on the QsnWTD, QsnReadInp, QsnReadMDT, and QsnReadMDTAlt APIs. These characters select specific operations for the display station to perform. Byte 1 is always processed first. When the CCs are used with the QsnWTD API, the first CC is processed immediately while the second CC is not processed until all the other information associated with the API has been processed. When used with an input operation, both CCs are processed after the operation has completed. The following two tables list the valid control character values and their associated mnemonics.

Control Character Byte 1						
Mnemonic	**Bits 0-2**	**Reset Pending Aid; Lock Keyboard**	**Clear Master MDT; Reset MDT Flags in Nonbypass Fields**	**Clear Master MDT; Reset MDT Flags in All Fields**	**Null Nonbypass Fields with MDT On**	**Null All Nonbypass Fields**
QSN_CC1_NULL	000					
QSN_CC1_LOCKBD	001	x				
QSN_CC1_MDTNBY	010	x	x			
QSN_CC1_MDTALL	011	x		x		
QSN_CC1_CLRMOD	100	x			x	
QSN_CC1_MDTNBY_CLRALL	101	x	x			x
QSN_CC1_MDTNBY_CLRMOD	110	x	x		x	
QSN_CC1_MDTALL_CLRALL	111	x		x		x
Note: 1. Bits 3 through 7 are reserved and must be 0. A CPFA31C error will be issued if this is not the case. 2. If there are no bypass fields with MDT flags on, then the master MDT will be cleared.						

Control Character Byte 2		
Mnemonic	**Bit**	**Meaning**
	0	reserved
QSN_CC2_NO_IC	1	0: Cursor moves to default or IC order position when keyboard unlocks 1: Cursor does not move when keyboard unlocks
QSN_CC2_RST_CSR_BL	2	0: no action 1: Reset blinking cursor
QSN_CC2_SET_CSR_BL	3	0: no action 1: Set blinking cursor
QSN_CC2_UNLOCKBD	4	0: no action 1: Unlock the keyboard and reset any pending AID bytes
QSN_CC2_ALARM	5	0: no action 1: Sound alarm
QSN_CC2_MSG_OFF	6	0: no action 1: Set Message Waiting indicator off
QSN_CC2_MSG_ON	7	0: no action 1: Set Message Waiting indicator on
Notes: • The mnemonics for control character byte 2 can be combined with a bitwise OR operation. • See notes in the 5250 data stream documentation for further details regarding these functions.		

Screen Attribute Characters

The screen or field attributes control the image produced on the display station screen. Each attribute occupies one character position in the display station regeneration buffer and is displayed as a blank. The effect produced by an attribute begins at its location in the regeneration buffer and continues until the next attribute appears. The attributes for non-color displays are shown in the table below and for color displays in the Screen Attributes for Color Displays (page 296) table. There are certain operations that allow a value to be specified for a screen attribute that indicates no screen attribute should be used. Where supported, the value is X'00' and the mnemonic is QSN_NO_SA.

Screen Attributes for Non-Color Displays

Mnemonic	Bits	Value
QSN_SA_NORM	0-2	001: Attribute identification flag
QSN_SA_CS	3	0: Column separator off 1: Column separator on
QSN_SA_BL	4	0: Do not blink field 1: Blink field
QSN_SA_UL	5	0: Do not underscore field 1: Underscore field
QSN_SA_HI	6	0: Low intensity 1: High intensity
QSN_SA_RI	7	0: Normal image 1: Reverse image
QSN_SA_ND		Non-display: equivalent to specifying QSN_SA_UL, QSN_SA_HI, and QSN_SA_RI.
Note: Multiple functions can be selected by combining the mnemonics with a bitwise OR operation.		

Screen Attributes for Color Displays

Mnemonic	Value	Meaning
QSN_SA_GRN	x'20'	Green
QSN_SA_GRN_RI	x'21'	Green/Reverse Image
QSN_SA_WHT	x'22'	White
QSN_SA_WHT_RI	x'23'	White/Reverse Image
QSN_SA_GRN_UL	x'24'	Green/Underscore
QSN_SA_GRN_UL_RI	x'25'	Green/Underscore/Reverse Image
QSN_SA_WHT_UL	x'26'	White/Underscore
QSN_SA_ND	x'27'	Nondisplay
QSN_SA_RED	x'28'	Red
QSN_SA_RED_RI	x'29'	Red/Reverse Image
QSN_SA_RED_BL	x'2A'	Red/Blink
QSN_SA_RED_RI_BL	x'2B'	Red/Reverse Image/Blink
QSN_SA_RED_UL	x'2C'	Red/Underscore
QSN_SA_RED_UL_RI	x'2D'	Red/Underscore/Reverse Image
QSN_SA_RED_UL_BL	x'2E'	Red/Underscore/Blink
QSN_SA_ND_2F	x'2F'	Nondisplay

Screen Attributes for Color Displays		
Mnemonic	**Value**	**Meaning**
QSN_SA_TRQ_CS	x'30'	Turquoise/Column Separators
QSN_SA_TRQ_CS_RI	x'31'	Turquoise/Column Separators/Reverse Image
QSN_SA_YLW_CS	x'32'	Yellow/Column Separators
QSN_SA_YLW_CS_RI	x'33'	Yellow/Column Separators/Reverse Image
QSN_SA_TRQ_UL	x'34'	Turquoise/Underscore
QSN_SA_TRQ_UL_RI	x'35'	Turquoise/Underscore/Reverse Image
QSN_SA_YLW_UL	x'36'	Yellow/Underscore
QSN_SA_ND_37	x'37'	Nondisplay
QSN_SA_PNK	x'38'	Pink
QSN_SA_PNK_RI	x'39'	Pink/Reverse Image
QSN_SA_BLU	x'3A'	Blue
QSN_SA_BLU_RI	x'3B'	Blue/Reverse Image
QSN_SA_PNK_UL	x'3C'	Pink/Underscore
QSN_SA_PNK_UL_RI	x'3D'	Pink/Underscore/Reverse Image
QSN_SA_BLU_UL	x'3E'	Blue/Underscore
QSN_SA_ND_3F	x'3F'	Nondisplay

Display Address

The display address is the address at which data is displayed or a field definition begins. This can be modified explicitly with a Set Output Address (QsnSetOutAdr) call, or implicitly with output operations, such as those associated with the Write Data (QsnWrtDta) API, that accept a cursor position. The 5250 Write to Display (WTD) command initializes the display address to row 1, column 1. Because each output operation contains a WTD command, this means that the display address is reset on each direct screen output operation.

Insert Cursor Address

The insert cursor (IC) order specifies the position of the cursor when the host system unlocks the keyboard and when the display station operator presses the Home key. The display address is not affected by this address. This can be set with the Insert Cursor (QsnInsCsr) API, and in some cases with the Set Cursor Address (QsnSetCsrAdr) API (only when the Move Cursor (MC) order is not supported).

Modified Data Tag (MDT) Bit

There is a modified data tag (MDT) bit for each input field and a master MDT bit. These bits are used to determine which fields should be returned in response to the Read Modified Fields (QsnReadMDT), Read Modified Alternate (QsnReadMDTAlt), and Read Modified Immediate Alternate (QsnReadMDTImmAlt) APIs. The MDT bit for a field and the master MDT bit can be set using bit 4 of the field format word (see "Format of the Field Format Word" on page 140) on a Set Field (QsnSetFld) API. The master MDT bit and the MDT bit for a field are set on anytime the operator types into or alters a field on the display. Once the bits are set, only a control character for resetting them (see Table 2), or a clear screen operation using the Clear Screen (QsnClrScr) API or a Start of Header order, can reset them.

Resequencing

Resequencing allows the control unit to return up to 128 input fields in any specified order. Resequencing is accomplished by chaining input fields together with Field Control Words specifying resequencing. (See "Format of the Field Control Word" on page 143 and the 5250 data stream documentation for details.)

States and Modes

The display station can be in one of several states (conditions), each with its accompanying modes (methods of operation). The following is a list of these states and their associated modes:

- Hardware error state
- Normal locked state
- Normal unlocked state
  - Command mode
  - Insert mode
  - Data mode
- Power-on state
- Prehelp error state
- Post-help error state
- System services (SS) message state
- System request state

See the 5250 data stream documentation for a detailed explanation of each state and mode.

Dumping the 5250 Data Stream Commands

If you wish to produce a dump of the 5250 data stream commands that are produced by the DSM APIs, you should create a physical file (using the CRTPF command) having a record length of 2000. Name the physical file QSNDEBUGF, and ensure that the QSNDEBUGF file exists in the library list. DSM will dump the 5250 data stream commands to that file.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

---

Using Window Services APIs

A window is created using a window and a low-level environment description. The window description provides the window attributes, a pointer to data that is specified by the using application, and several exit routines that the window module calls when a window is moved, resized, or deleted so the using program can perform the appropriate actions. The low-level environment description is the same as that used on the Create Low-Level Environment (QsnCrtEnv) API to create a low-level environment. A window is implemented as a low-level environment where the user data pointer describes the window itself. Thus, a window can be manipulated through the low-level APIs or through the window APIs by using the same window handle. This implementation is similar to the concept of inheritance in object-oriented programming languages. DSM window support uses graphical user interface (GUI) when the underlying control unit supports it.

Each window has the low-level environment window mode enabled. The low-level environment window area is set to the usable area in the window, which consists of the area inside the border and attributes that can be accessed by screen I/O services. (It does not include the message line.) Use relative coordinates when specifying a row and column on an I/O API. The upper left corner of the usable area is (1,1). To use absolute coordinates with a window, disable the low-level environment window mode with the Set Low-Level Environment Window Mode (QsnSetEnvWinMod) API.

Figure 1 (page 298) shows the components of a DSM window. The window in this example has a specified depth of 13 rows and a width of 19 columns.

The attributes of a DSM window are similar to those of a data description specifications (DDS) window. The initial size and location of a DSM window are specified using the location of the upper-left window border character and the number of rows and columns within the window. For DSM windows, the

leading window attribute, right continuation attribute, or message line can be specified separately. Unlike a DDS window, a DSM window does not require the following:

- A border

  If a window is defined with no border, no extra space is used on the display for the border characters or their attributes (L and B in Figure 1 (page 298)). An area of the screen is cleared starting from the specified location for the upper left corner of the window and continues for the number of rows and columns given as the window size. Figure 2 (page 299) shows an example of a window with no border.

  Note: In discussions throughout the DSM sections that refer to window borders, this should be taken to mean the top/bottom window row or left/right window column for a window with no borders.

- Window border attributes

  If a window is defined with no border attributes (L and B in Figure 1 (page 298)), no extra space for these is used on the display. Figure 3 (page 300) shows a window with no border attributes.

- Leading window attribute

  The leading window attribute (A in Figure 1 (page 298)) is part of the addressable window area. If specified, this attribute takes up an extra screen character and does not reduce the size of the window area. Figure 4 (page 300) shows a window defined with no leading window attribute. The window text directly follows the window border character. Attribute characters can be written inside the window if desired.

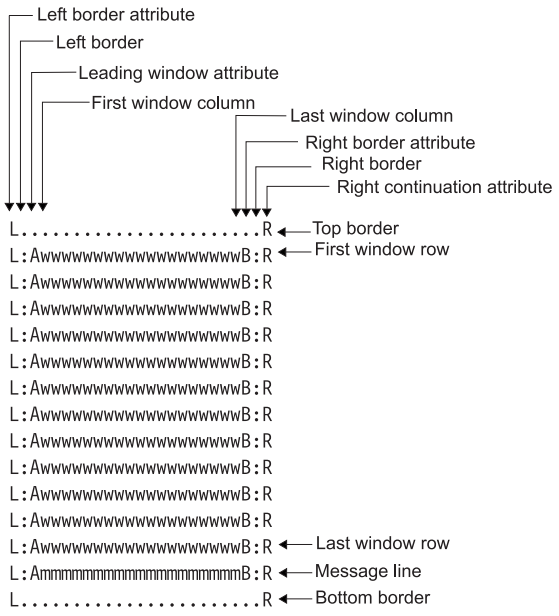
- Right continuation attribute

  If the right continuation attribute (R in Figure 1 (page 298)) is specified on a window description, DSM determines the appropriate attribute to use, based on the screen image underlying the window. If the window is not a GUI window, right-continuation-attribute correction is performed for display-screen and presentation screen DBCS-only attributes. (No correction is performed for extended primary attributes and DBCS data types). When a window is placed on a screen that supports extended attributes, all extended attributes are cleared prior to displaying the window. To have the right continuation attribute handle extended attributes, you must use a display that supports GUI windows and specify GUI window support on the window description.

- Message line

  Specifying a message line on a window description decreases the number of lines in the window area by 1, as shown in Figure 1.

Figure 1. DSM Window Layout



Key

L	Left border attribute
.	Top and bottom border
R	Right continuation attribute
:	Left and right border
A	Leading window attribute
w	Window area
m	Message line
B	Right border attribute

Figure 2. DSM Window with No Border

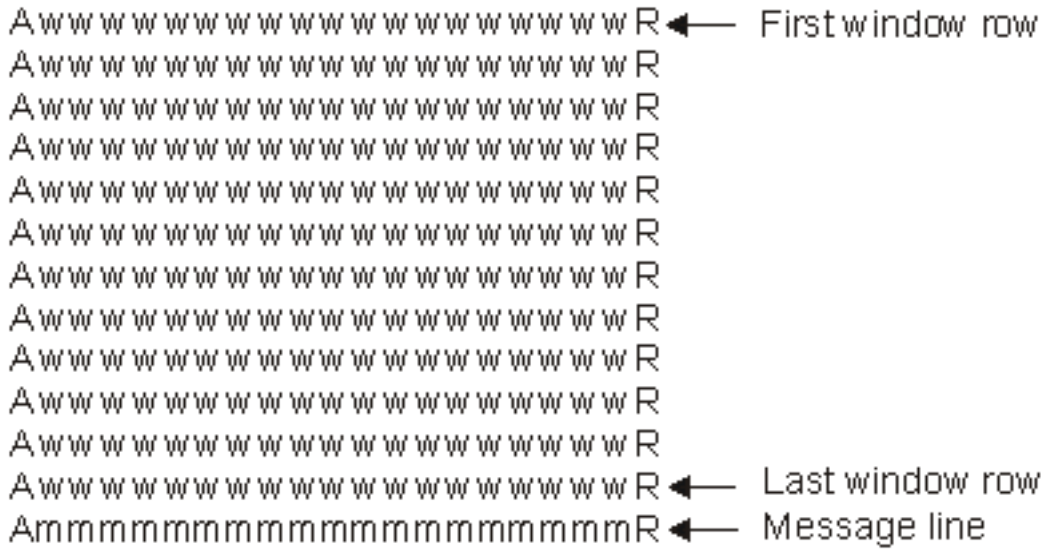


Figure 3. DSM Window with No Border Attributes

Figure 4. DSM Window with No Leading Window Attribute

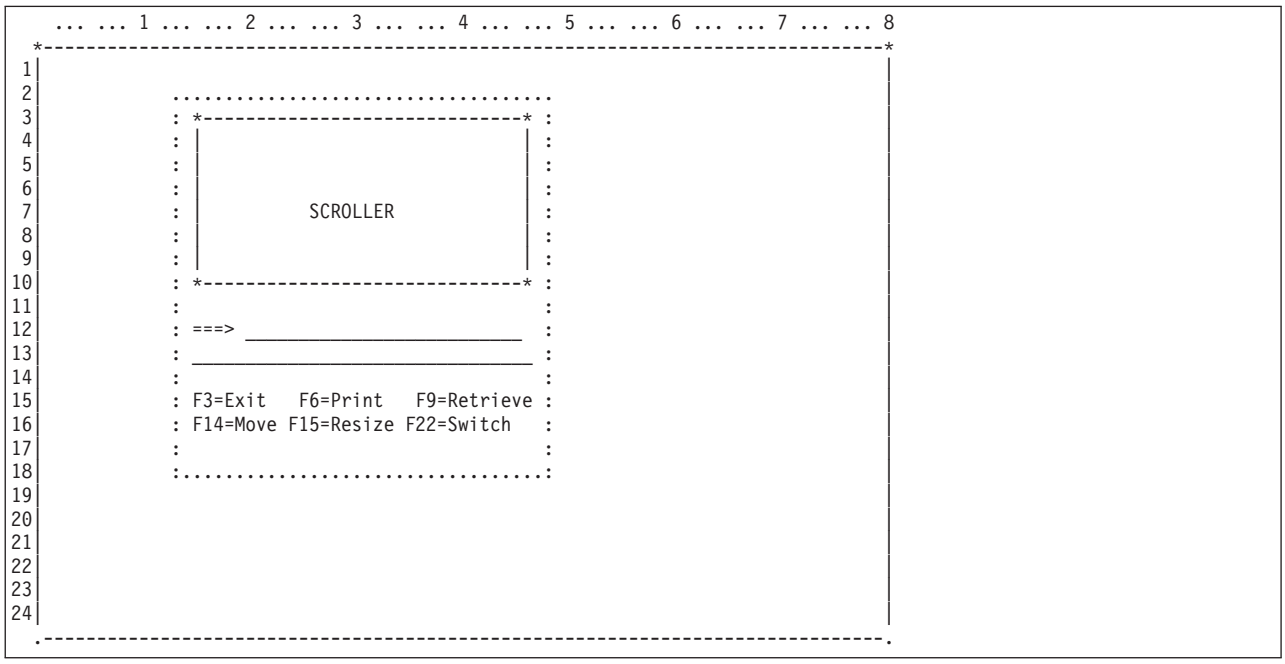
Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Using Session Services APIs

Session Details

The following figure shows the default attributes provided by the DSM session description for a session.

Session Attributes



The main component of a session is the scrollable area, or **scroller**, where output data can be displayed for the session. A session may or may not have a data input line, depending on the application. A session that uses the default attributes has an input line underneath the scroller. You can allow the size and location of the session attributes to default based on the window size, or you can specify these explicitly. Up to two lines of command key descriptions can appear below the scroller and can be managed by a session. For details on the session description see "Create a Session (QsnCrtSsn) API" on page 227.

When a window containing a session is moved or resized, the scroller and any automatically defined fields are redrawn to reflect the new window positions and size. If any additional items have been added to the session through the low-level interface APIs, you must supply an exit program that will reposition such items explicitly. See "Create a Session (QsnCrtSsn) API" on page 227 for details on the exit program.

Line Mode and Character Mode I/O

Session I/O can be performed in a line mode or a character mode basis. In line mode, each call to the line-specific interfaces operates on a complete line, either on input or output. In character mode, I/O is performed a character at a time. This means that multiple I/O operations can be issued to operate on the current line. For example, an output operation could output several characters. Then a backspace operation could be followed by input from the current cursor position. (All input operations are still performed in block mode, where the input is not available until an AID-generating key has been pressed.)

Line mode output is performed using the Write Line to Scroller (QsnWrtSclLin) API. This API writes a line of data to the next session line and sets the active position (see "Active Position" on page 304) to the start of the next line after the added line. For character output, the Write Characters to Scroller (QsnWrtSclChr) API is used. This API outputs a string of characters starting at the active position. After this operation completes, the active position is one position past the last character written, or it is the position specified by a control character sequence if this appears at the end of the data.

Command Key Action Routines

Part of the session description is an array of command key actions. Each action is an exit routine that is specified as a function pointer. When a command key is pressed during a QsnReadSsnDta operation, if an action has been specified, the appropriate exit routine is called. Otherwise, an `Invalid key pressed` error message will be issued. DSM provides a group of functions that can be called, or user-defined exit-routines can be specified. The action routines are specified as an array of 24 function pointers in the session description. (See "Create a Session (QsnCrtSsn) API" on page 227 for details.) The default values for the action routines DSM calls are:

Cmd Key	Action Routine
1	
2	
3	
4	
5	
6	Print Scroller Data (QsnPrtScl)
7	Roll Scroller Down (QsnRollSclDown)
8	Roll Scroller Up (QsnRollSclUp)
9	Retrieve Session Input Line to Input Line (QsnRtvSsnLin)
10	
11	Toggle Line Wrap/Truncate Mode (QsnTglSclWrp)
12	
13	
14	Move Window by User (QsnMovWinUsr)
15	Resize Window by User (QsnRszWinUsr)
16	
17	Display Scroller Top (QsnDspSclT)
18	Display Scroller Bottom (QsnDspSclB)

Cmd Key	Action Routine
19	Shift Scroller Left (QsnShfSclL)
20	Shift Scroller Right (QsnShfSclR)
21	Display Command Line Window (direct mapping to QUSCMDLN API)
22	
23	
24	

The default action routines for command keys 7, 8, 19, and 20 (QsnRollSclDown, QsnRollSclUp, QsnShfSclL, and QsnShfSclR, respectively) will pass any numeric input to the API when the command key is pressed. For example, to shift the scroller to the right by 10 columns, the value 10 could be entered at the input line prior to pressing command key 20. Non-numeric input is ignored.

When a user-defined action routine is called, it is passed the following information:

Information Passed to the Action Routine	
1	Session handle
Input	Binary(4)
2	Input buffer
Input	Binary(4)
3	Returned action
Output	Char(1)

When the specified command key is pressed, the action routine for the command key is called. If you change the default values to have a command key call a different DSM API, you cannot specify the API directly because the action routine is passed specific parameters. You must define an action routine that can accept the action routine parameters and then call the desired DSM API with the appropriate parameters. You can define a generic action routine that is specified for each key you want to define, and in that action routine query the input buffer to determine the command key pressed and the appropriate action to take.

When an action routine is called, any data on the input line will remain. You can use the QsnWTD API to clear the line. However, if you write to the session or perform any action that causes the session to be redrawn in the action routine, the data on the input line will be lost.

If an exception occurs during the processing of an action routine, it is ignored and processing continues. A CPFA3D9 will be issued as an exception from the QsnReadSsnDta API when control returns from the action routine. You can handle exceptions explicitly by adding an exception handler to the action routine.

Action Routine Parameters

Session handle

INPUT; BINARY(4)

The session currently active. If the action routine causes the active session to change, this variable will be changed to reflect the new session.

Input buffer

INPUT; BINARY(4)

The input buffer containing the results of the input operations that caused this exit routine to be called. The input buffer can be queried using the low-level interface routines. This is the buffer that was passed to QsnReadSsnDta.

Returned action
OUTPUT; CHAR(1)

The variable containing the flag indicating if, following a successful return from this exit routine, control returns to the caller of the Read Data from Session (QsnReadSsnDta) API or if QsnReadSsnDta handles the next input operation. If an error occurs in the exit routine, control always returns to the caller. The possible values are:

0	QsnReadSsnDta continues to handle the next input operation. Control does not return to the caller.
1	QsnReadSsnDta returns control to the caller. The output parameters for QsnReadSsnDta are filled in appropriately.

Active Position

The active position in the scroller is the point at which data will be written for character mode operations. The active position is affected by output operations to the scroller, including the writing of data that contains EBCDIC display control character sequences.

» DSM does not allow the application to partially overwrite data of one type, with another type of data. Instead, it removes the mismatched data and all data that follows it from the scroller line, so that the new data may be inserted in its place.

A single scroller line can contain a mixture of data types, including EBCDIC, DBCS and CCSID-based data. The QsnSclBS and QsnSclCR APIs change the active position of the scroller line. Subsequent calls to the QsnWrtSclChr or QsnWrtSclChrCC APIs will cause existing data in the line to be overwritten, starting at the active position. If the new data is the same type or CCSID of the existing data, the new data will replace the existing data. However, if the existing data is not the same type or CCSID as the new data, DSM will remove the subset of mismatched existing data from the line. Any data in the line that follows the mismatched data will also be removed.

For example, the scroller line contains some EBCDIC data followed by DBCS data. The QsnSclBS API is called such that the active position is moved within the DBCS data. If QsnWrtSclChr is called to write EBCDIC data to the active position, the DBCS data is removed from the line and the new EBCDIC data is written. The result is a line that contains the original EBCDIC data with the new EBCDIC data appended to it.

Consider the same scroller line that contains EBCDIC followed by DBCS data. QsnSclBS is called such that the active position is before the last byte of EBCDIC data. If QsnWrtSclChr is called to write two bytes of EBCDIC data, the last byte of existing EBCDIC data is replaced. However, the second byte of the new data would overwrite the shift out character of the existing DBCS data. DSM will remove the DBCS data from the line, and write the last byte of the new EBCDIC data to the line. The result is a line that contains all but the last byte of the original EBCDIC data with the new EBCDIC data appended to it.

In both examples, if other combinations of EBCDIC, DBCS or CCSID-based data were present in the line after the DBCS data, all of that data would have also been removed with the DBCS data. If other combinations of data types or CCSIDs in the line would have preceded the EBCDIC data, all of the preceding data would be unchanged. «

EBCDIC Display Control Characters

The data written to the scroller may contain display control characters consisting of single byte EBCDIC values. If specified on the session description (see “Create a Session (QsnCrtSsn) API” on page 227), the APIs for writing data to the scroller will check for and interpret such control characters. Each control

character recognized in the output data is replaced by a call to a DSM API or internal routine that will perform the appropriate function. The following table shows the control characters that are recognized and the APIs that are called, where applicable.

EBCDIC Display Control Characters

Character	Hex Value	Interpretation
HT	05	QsnSclTab
VT	0B	QsnSclLF
FF	0C	QsnSclFF
CR	0D	QsnSclCR
NL	15	QsnSclNL
LF	25	QsnSclNL
BS	16	QsnSclBS
BEL	2F	QsnBeep

DBCS Considerations

If the low-level environment description (see "Format of the Low-Level Environment Description" on page 10) for the session specifies DBCS support, the session services will check for and handle DBCS data. DBCS data must be enclosed by shift control (SO/SI) characters. The DBCS support field determines the type of the input field defined for the session, but does not affect the checking done for session output data other than to indicate that DBCS data may be present. The scroller does not display data using extended NLS attributes, regardless of the underlying display device support.

Top | "Dynamic Screen Manager APIs," on page 1 | APIs by category

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

© Copyright IBM Corp. 1998, 2005

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI

DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for downloading and printing publications

Permissions for the use of the information you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce this information for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of this information, or any portion thereof, without the express consent of IBM(R).

Commercial Use: You may reproduce, distribute and display this information solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of this information, or reproduce, distribute or display this information or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the information or any data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the information is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THIS INFORMATION. THE INFORMATION IS PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

All material copyrighted by IBM Corporation.

By downloading or printing information from this site, you have indicated your agreement with these terms and conditions.

Code disclaimer information

This document contains programming examples.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM(R), ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Printed in USA