Basic Programming
Reference Manual
## Contents

About this Manual ................................................................. xxi

**Section 1 Introduction**

ICS Accessory API ............................................................. 1–2
ICS Library API ................................................................. 1–2

**Section 2 Functions By Category**

ICS Accessory API ............................................................. 2–1
  - MS-Windows API .......................................................... 2–2
  - XVT/Win API .............................................................. 2–3
ICS DosLink Client/Server Applications............................. 2–5
ICS Memory Management API .......................................... 2–6
  - MS-Windows API .......................................................... 2–6
  - XVT/Win API .............................................................. 2–6
General Utilities ............................................................... 2–7
  - MS-Windows API .......................................................... 2–8
  - XVT/Win API .............................................................. 2–9
ICS Library API ................................................................. 2–10
  - Entry Points Provided by SLs and EILs ..................... 2–10
    - MS-Windows API .......................................................... 2–10
  - ICS Utilities for Library Development ..................... 2–12
    - MS-Windows API .......................................................... 2–12

**Section 3 INFOConnect API**

- IcAddRefContextID.......................................................... 3–2
- IcAllocBuffer .............................................................. 3–3
- IcChangeHandle ........................................................... 3–4
- IC_CHECK_DATAFLAGS .............................................. 3–6
- IC_CHECK_RESULT_SEVERE ........................................ 3–7
- IcCloseSession ........................................................... 3–8
- IcCreateHandle ........................................................... 3–9
- IcCreateHwnd ............................................................. 3–10
- IcCreateSession ......................................................... 3–11
- IcDefaultErrorProc ..................................................... 3–12
Contents

IcLibSetResult .................................................................3–71
IcLibTerminate ...............................................................3–72
IcLibUpdateConfig .......................................................3–73
IcLibVerifyConfig .........................................................3–76
IcLibXmt .................................................................3–78
IcLockBuffer ............................................................3–79
IC_MAKE_RESULT .........................................................3–80
IcMgrEilEvent ...............................................................3–81
IcMgrGetSessionInfo ...................................................3–82
IcMgrLcl .................................................................3–83
IcMgrRcv ...............................................................3–84
IcMgrSendEvent ..........................................................3–85
IcMgrSetResult ............................................................3–87
IcMgrTraceBuffer .........................................................3–88
IcMgrTraceResult ..........................................................3–90
IcMgrXmt .................................................................3–92
IcNextEvent .................................................................3–93
IcNotifyConfig ..............................................................3–95
IcOpenAccessory ........................................................3–97
IcOpenSession ..............................................................3–100
IcRcv .................................................................3–104
IcReadBuffer ...............................................................3–106
IcReadLibraryConfig ..................................................3–107
IcReAllocBuffer ..........................................................3–109
IcRegisterAccessory ......................................................3–110
IcRegisterCallback .......................................................3–111
IcRegisterMsgSession ..................................................3–113
IcReleaseContextID ....................................................3–115
IcRunAccessory ............................................................3–116
IcRunHelp3 .................................................................3–118
IcRunLibHelp ...............................................................3–120
IcSelectPath .................................................................3–121
IcSetError .................................................................3–123
IcSetServerInfo ...........................................................3–124
IcSetSessionError .......................................................3–125
IcSetStatus .................................................................3–127
IcUnlockBuffer ............................................................3–128
IcWriteBuffer ...............................................................3–129
IcWriteLibraryConfig ..................................................3–131
IcXmt .................................................................3–133
NOREF .................................................................3–135
ic_buf_alloc ...............................................................3–136
ic_buf_free ...............................................................3–137
ic_buf_lock ...............................................................3–138
ic_buf_realloc ............................................................3–139
Contents

ic_buf_unlock ................................................................. 3–140
ic_change_handle .......................................................... 3–141
ic_close_session ............................................................ 3–142
ic_default_error_proc .................................................... 3–143
ic_deregister_accessory ............................................... 3–145
ic_exit_ok ........................................................................ 3–146
ic_galloc ........................................................................ 3–147
ic_get_context ............................................................... 3–148
ic_get_context_string.................................................... 3–149
ic_get_infoconnect_dir ................................................ 3–150
ic_get_new_path ............................................................ 3–151
ic_get_path_id ............................................................... 3–153
ic_get_path_names ......................................................... 3–154
ic_get_session_id .......................................................... 3–155
ic_get_session_info ...................................................... 3–156
ic_get_string ............................................................... 3–157
ic_gfree .......................................................................... 3–158
ic_glock ......................................................................... 3–159
ic_grealloc ....................................................................... 3–160
ic_gunlock ....................................................................... 3–161
ic_init_ics ....................................................................... 3–162
ic_lcl ............................................................................... 3–163
ic_open_accessory ........................................................ 3–164
ic_open_session ............................................................ 3–167
ic_rcv .............................................................................. 3–170
ic_register_accessory ................................................... 3–171
ic_register_msg_session .............................................. 3–172
ic_run_accessory .......................................................... 3–174
ic_set_error ..................................................................... 3–176
ic_set_status ................................................................. 3–177
ic_xmt .............................................................................. 3–178

Section 4 ICS Messages/Events

E_IC_ERROR ...................................................................... 4–3
E_IC_LCL_RESULT............................................................ 4–4
E_IC_NEWPATH ............................................................... 4–5
E_IC_NULLEVENT .......................................................... 4–6
E_IC_RCV_DONE ............................................................. 4–7
E_IC_RCV_ERROR........................................................... 4–8
E_IC_SESSION_CLOSE .................................................. 4–9
E_IC_SESSION_EST ......................................................... 4–10
E_IC_STATUS.................................................................... 4–11
E_IC_STATUS_RESULT .................................................. 4–12
E_IC_XMT_DONE ............................................................ 4–13
Section 5 ICS Data Structures/Types

CHANNELID ................................................................. 5–1
EVENT ............................................................................. 5–2
HIC_CHANNEL ................................................................. 5–3
HIC_CONFIG ................................................................. 5–3
HIC_SESSION ................................................................. 5–3
HIC_STATUSBUF ............................................................. 5–4
IC_BASEREVISION ........................................................... 5–4
IC_BASEVERSION ............................................................ 5–4
IC_BUFINDD ................................................................. 5–5
IC_BUILD_REVISION .......................................................... 5–5
IC_CALLBACK ................................................................. 5–5
IC_COMMAND ................................................................. 5–6
IC_COMPONENT .............................................................. 5–7
IC_COMPONENT_TYPE ....................................................... 5–7
IC_DEBUG ..................................................................... 5–10
IC_DICT_FIELD ............................................................... 5–12
IC_DICT_NODE ............................................................... 5–13
IC_DICT_TABLE ............................................................... 5–15
IC_DIRECTORYTYPES ...................................................... 5–16
IC_EMU_LEVEL ............................................................... 5–17
IC_ERROR_INFO ............................................................. 5–17
IC_ERROR_MASK ............................................................. 5–18
IC_ERROR_SEVERE .......................................................... 5–18
IC_ERROR_TERMINATE ...................................................... 5–19
IC_ERROR_WARNING ....................................................... 5–19
Contents

IC_FIELD_FLAGS............................................................. 5–20
IC_FIELDTYPE.................................................................. 5–22
IC_HEADER_SIZE ............................................................ 5–25
IC_HEADER_3_0 .............................................................. 5–25
IC_KEY_SERIALNUM....................................................... 5–26
IC_LCL_FLAGS .............................................................. 5–26
IC_LIBRARY_FLAGS ....................................................... 5–27
IC_MAXACCESSORYIDLEN............................................ 5–27
IC_MAXACCESSORYIDSIZE........................................... 5–28
IC_MAXCHANNELIDLEN ................................................ 5–28
IC_MAXCHANNELIDSIZE ................................................. 5–28
IC_MAXCONNECTEDPATHIDLEN .................................. 5–28
IC_MAXDESCRIPTIONSIZE ............................................. 5–29
IC_MAXERRORINSERT ................................................... 5–29
IC_MAXERRORSTRING ................................................... 5–29
IC_MAXFILENAMESIZE ................................................... 5–30
IC_MAXIDSIZE .............................................................. 5–30
IC_MAXLIBRARYIDLEN................................................... 5–30
IC_MAXLIBRARYIDSIZE.................................................. 5–31
IC_MAXPACKAGEIDSIZE............................................... 5–31
IC_MAXPATHIDLEN ........................................................ 5–31
IC_MAXPATHIDSIZE ....................................................... 5–31
IC_MAXPRINTSTRING ..................................................... 5–32
IC_MAXSESSIONIDLEN ................................................ 5–32
IC_MAXSESSIONIDSIZE .................................................. 5–32
IC_MAXSESSIONIDSUFFIX ............................................. 5–33
IC_MAXSTRINGLENGTH ................................................. 5–33
IC_MAXTEMPLATEIDLEN ............................................... 5–33
IC_MAXTEMPLATEIDSIZE............................................... 5–33
IC_MAXVENDORNAMELEN ............................................ 5–34
IC_MAXVENDORNAMESIZE............................................ 5–34
IC_MAXWSIDSIZE ........................................................ 5–34
IC_MEMHND ................................................................. 5–34
IC_MINOR_VERSION ....................................................... 5–35
IC_MSG_CONFIG ............................................................. 5–35
IC_NEXTEVENT_FLAGS.................................................. 5–37
IC_OK ................................................................................ 5–38
IC_OPEN_OPTIONS ......................................................... 5–38
IC_PACKAGE................................................................. 5–39
IC_PATH_FLAGS.............................................................. 5–39
IC_PRINT_SUMMARY ...................................................... 5–39
IC_RC_NODE ................................................................. 5–40
IC_RECORD_INFO ........................................................... 5–43
IC_RECORD_SIZE ........................................................... 5–44
IC_RESULT ....................................................................... 5–44
IC_RESULT_CONTEXT_CFG...........................................5–45
IC_RESULT_CONTEXT_ICDB..........................................5–45
IC_RESULT_CONTEXT_ICUTIL.......................................5–45
IC_RESULT_CONTEXT_INVALID ....................................5–46
IC_RESULT_CONTEXT_STD ...........................................5–46
IC_RESULT_SUBTYPE.....................................................5–46
IC_RESULT_SUBVALUE..................................................5–47
IC_REVISION...................................................................5–47
IC_REVISIONNUM.............................................................5–48
IC_SERIALNUM.................................................................5–48
IC_SESSION_FLAGS........................................................5–49
IC_SINFO ...........................................................................5–50
IC_STATUSBUF............................................................5–50
IC_STATUS_BLOCKING ..................................................5–54
IC_STATUS_BUFFER.......................................................5–55
IC_STATUS_COMMGR..................................................5–56
IC_STATUS_CONNECT....................................................5–57
IC_STATUS_CONTROL....................................................5–59
IC_STATUS_DATAFLAGS ...............................................5–60
IC_STATUS_FKEY............................................................5–62
IC_STATUS_LINESTATE..................................................5–63
IC_STATUS_REACTIVATE.............................................5–64
IC_STATUS_TRANS.........................................................5–65
IC_STATUS_UTS............................................................5–66
IC_TABLE_FLAGS............................................................5–66
IC_TABLETYPE.................................................................5–68
IC_TemplateBegin ............................................................5–69
IC_TemplateChannel.......................................................5–70
IC_TemplateConfig........................................................5–70
IC_TemplateConfigTable .................................................5–71
IC_TemplateDescription ..................................................5–71
IC_TemplateEnd ...............................................................5–72
IC_TemplateFlags.............................................................5–72
IC_TemplateInit.................................................................5–72
IC_TemplateLibrary ..........................................................5–74
IC_TemplateOpenID .........................................................5–74
IC_TemplateTerm .............................................................5–75
IC_UPGRADE_INFO .........................................................5–75
IC_VER...............................................................................5–77
IC_VER_INFO..................................................................5–77
IC_VERIFY .........................................................................5–78
IC_VERIFY_OK...............................................................5–79
IC_VERSION_FILE ............................................................5–79
IC_VERSION_PRODUCT ..................................................5–80
IC_VERSION_STRING....................................................5–80
Section 6  ICS Accessory Definition

Appendix A  Standard IDs (Keys) & Component Numbers

Accessory IDs .......................................................... A–1
Service Library IDs............................................... A–2
External Interface Library IDs ................................. A–3
Component Numbers............................................ A–4
Appendix B Status Types and Statuses

Statuses Sent from Accessory to Library ....................... B–2
  IC_STATUS_BLOCKING........................................... B–2
  IC_STATUS_BUFFER............................................ B–2
  IC_STATUS_CONNECT ......................................... B–4
  IC_STATUS_DATAFLAGS....................................... B–4
  IC_STATUS_FKEY............................................... B–5
  IC_STATUS_REACTIVATE..................................... B–6
  IC_STATUS_TRANS............................................. B–6

Statuses Sent from Library to Accessory ....................... B–7
  IC_STATUS_CONNECT ......................................... B–7
  IC_STATUS_CONTROL......................................... B–8
  IC_STATUS_LINESTATE....................................... B–9

Statuses Sent from Accessory to Accessory ................... B–10
  IC_STATUS_DATAFLAGS....................................... B–10
  IC_STATUS_CONTROL......................................... B–11

Statuses Sent from ICS to Accessory .......................... B–12
  IC_STATUS_COMMMGR......................................... B–12

UTS-Specific Statuses .............................................. B–14
  IC_STATUS_UTS................................................. B–14
    IC_UTS_SELECTION subtype 0............................ B–14
    IC_UTS_DVC_READY subtype 0x10...................... B–15
    IC_UTS_DVC_BUSY subtype 0x11....................... B–15
    IC_UTS_DVC_ERROR subtype 0x12..................... B–15
    IC_UTS_DVC_NOTREADY subtype 0x13................. B–15
    IC_UTS_ATTENTION subtype 0x20.................... B–15

DosLink-Specific Statuses .......................................... B–16
  DOSLINK_SINFO................................................ B–16

Library Support for 1.11 Applications .......................... B–17
  UTS EIL (and INT1 SL).................................... B–17
    TTY EIL..................................................... B–17
  IC_STATUS_SPECIALMSG.................................... B–18
    TTY EIL..................................................... B–18
    From UTS EIL or INT1 SL to the Accessory........ B–18
    From Accessory to UTS EIL or INT1 SL............. B–18
Appendix C  Errors and Results

INFOConnect Connectivity Services................................. C–2
ICCS Standard Errors.................................................. C–2
IC_ASSIGNMENT_ERROR (Value 902)................................ C–3
IC_ASSIGNMENT_UPDATED (Value 2004)...................... C–3
IC_CANCELED (Value 2003)..................................... C–3
IC_COMPLETE (Value 2013)..................................... C–4
IC_CONTEXT_ALREADY_CREATED (Value 701).................... C–4
IC_CONTEXT_ALREADY_DELETED (Value 702)..................... C–4
IC_CONTEXT_INVALID (Value 703).......................... C–5
IC_CONTEXT_NOT_FOUND (Value 704)...................... C–5
IC_CONTEXTSTRING_NOT_FOUND (Value 705).................... C–5
IC_CONTEXTSTRING_TRUNCATED (Value 706)................... C–6
IC_CONTEXTTABLE_FULL (Value 700)............................ C–6
IC_ERROR_ACCESSORY_FAILED (Value 801)..................... C–6
IC_ERROR_ACCESSORY_NOT_FOUND (Value 800)............... C–7
IC_ERROR_ALREADYCLOSED (Value 509)......................... C–7
IC_ERROR_APP_BUSY (Value 11).............................. C–7
IC_ERROR_APP_GONE (Value 12)............................ C–8
IC_ERROR_BADFUNCTION (Value 300)....................... C–8
IC_ERROR_BADPARAMETER (Value 4)........................ C–8
IC_ERROR_BADREVISION (Value 302)..................... C–9
IC_ERROR_BADSESSION (Value 1).......................... C–9
IC_ERROR_BADTEMPLATE (Value 611).......................... C–10
IC_ERROR_BADVERSION (Value 301)........................ C–10
IC_ERROR_CANCELOPEN (Value 2000)........................ C–10
IC_ERROR_CHAN_BUSY (Value 612)......................... C–11
IC_ERROR_CHANNELINUSE (Value 503)................ C–11
IC_ERROR_COLON_PRESENT (Value 906)...................... C–11
IC_ERROR_INITICS (Value 500)............................ C–12
IC_ERROR_INMODIFY (Value 507).......................... C–12
IC_ERROR_INTERNAL (Value 5).............................. C–12
IC_ERROR_INVALID_CONFIGREC (Value 900).................. C–13
IC_ERROR_INVALIDPATH (Value 502)........................ C–13
IC_ERROR_INVALID_WINCOMBO (Value 8)..................... C–13
IC_ERROR_INVALID_WINOPTION
(Value 7).................................................. C–14
IC_ERROR_LIBRARY_CONFIG
(Value 901).............................................. C–14
IC_ERROR_LOSTRCV (Value 305)........... C–14
IC_ERROR_LOSTXMT (Value 306)............ C–15
IC_ERROR_MGR_BUSY (Value 9).............. C–15
IC_ERROR_NEWREVISION (Value 615)....... C–15
IC_ERROR_NEWVERSION (Value 605)......... C–16
IC_ERROR_NOCHANDATA (Value 609)....... C–16
IC_ERROR_NOCLOSE (Value 508)............ C–16
IC_ERROR_NODATABASE (Value 102)........ C–17
IC_ERROR_NOFIN (Value 2008).............. C–17
IC_ERROR_NOLIBLOAD (Value 600)......... C–17
IC_ERROR_NOLIBRARY (Value 607)......... C–18
IC_ERROR_NOMEMORY (Value 3)............. C–18
IC_ERROR_NOPARTNER (Value 303)......... C–18
IC_ERROR_NOPATHDATA (Value 608)....... C–19
IC_ERROR_NOPATHID (Value 903)........... C–19
IC_ERROR_NORCVMEM (Value 309)......... C–19
IC_ERROR_NOSESSION (Value 2001)....... C–20
IC_ERROR_NOSESSIONMEM (Value 307).... C–20
IC_ERROR_NOTEMPLATE (Value 610)........ C–20
IC_ERROR_NOVERSION (Value 603)......... C–21
IC_ERROR_NOXMTMEM (Value 308)......... C–21
IC_ERROR_OLDVERSION (Value 614)........ C–21
IC_ERROR_PATHBUSY (Value 510).......... C–22
IC_ERROR_PATHID_EXISTS (Value 908).... C–22
IC_ERROR_PICCHANLINUSE
(Value 504)............................................ C–22
IC_ERROR_PIVERSION (Value 602)......... C–23
IC_ERROR_PMCHANLINUSE
(Value 505)............................................ C–23
IC_ERROR_PMVERSION (Value 601)......... C–23
IC_ERROR_QUEUEFULL (Value 904)......... C–24
IC_ERROR_RCV_BUSY (Value 10).............. C–24
IC_ERROR_REOPEN (Value 2).............. C–24
IC_ERROR_SERVICE_NOT_AVAILABLE
(Value 1001)........................................... C–25
IC_ERROR_SHELL_ACTIVE (Value 103)...... C–25
IC_ERROR_SIZE_EXCEEDED
(Value 904)............................................ C–25
IC_ERROR_SPACE_PRESENT
(Value 905)............................................. C–26
IC_ERROR_TERMINATE_CLEAR  
(Value 104) ............................................. C–26

IC_ERROR_TERMINATE_EXIT  
(Value 105) ............................................. C–27

IC_ERROR_TERMINATE_NOMSG  
(Value 0) ................................................. C–27

IC_ERROR_TERMINATE_SHUTDOWN  
(Value 106) ............................................. C–28

IC_ERROR_TILDE_PRESENT  
(Value 907) ............................................. C–28

IC_ERROR_TIMERS (Value 1) ....................... C–28

IC_ERROR_TRUNCATED (Value 2002) ........ C–29

IC_ERROR_UNIMPLEMENTED  
(Value 2012) ........................................... C–29

IC_ERROR_UNKNOWN (Value 1000) ........... C–29

IC_ERROR_UNKNOWN_COMMAND  
(Value 2010) ........................................... C–30

IC_ERROR_UNKNOWN_PARAMETER  
(Value 2009) ........................................... C–30

IC_ERROR_UNKNOWN_TABLE  
(Value 2011) ........................................... C–30

IC_ERROR_UNOPENEDSESSION  
(Value 506) ............................................. C–31

IC_ERROR_UPGRADE_WAIT (Value 613) ... C–31

IC_ERROR_WRONGVERSION  
(Value 604) ............................................. C–31

IC_ERROR_XMT_BUSY (Value 6) ................. C–32

IC_IGNORE (Value 2007) ............................... C–32

IC_INCOMPLETE (Value 2006) ...................... C–32

IC_INFO_QEVENT (Value 320) ...................... C–33

IC_OK (Value 0) .............................................. C–33

IC_VERIFY_OK (Value 2005) ......................... C–33

ICS Standard Configurator Errors ........................... C–34

IC_CFG_ALREADY_ACTIVE (Value 141) ...... C–34

IC_CFG_BIT_FIELD (Value 134) ................. C–34

IC_CFG_DATA_MISMATCH (Value 113) ........ C–34

IC_CFG_DATA_TRUNCATED (Value 133) .... C–35

IC_CFG_DELETE_INUSE (Value 143) ........ C–35

IC_CFG_DIFFERENT_ACTIVE (Value 140) .. C–35

IC_CFG_INFO_EXCESS (Value 132) .......... C–36

IC_CFG_INFO_IMPOSSIBLE (Value 127) ..... C–36

IC_CFG_INFO_TRUNCATED (Value 131) .... C–36

IC_CFG_INTERNAL_ERROR (Value 100) ...... C–36

IC_CFG_INVALID_DATABASE (Value 160) .. C–37

IC_CFG_INVALID_DB (Value 105) ............... C–37
Contents

IC_CFG_INVALID_DBMODE (Value 106)........ C–37
IC_CFG_INVALID_FIELD (Value 109).......... C–38
IC_CFG_INVALID_FIELD_TYPE
  (Value 111)........................................... C–38
IC_CFG_INVALID_HANDLE (Value 103)....... C–38
IC_CFG_INVALID_HWND (Value 161)......... C–39
IC_CFG_INVALID_KEY (Value 108)........... C–39
IC_CFG_INVALID_LIBRARY (Value 104)...... C–39
IC_CFG_INVALID_POSITION (Value 112)..... C–40
IC_CFG_INVALID_PROPERTY
  (Value 116)........................................... C–40
IC_CFG_INVALID_SIZE (Value 114).......... C–40
IC_CFG_INVALID_TABLE (Value 107)......... C–41
IC_CFG_INVALID_TABLE_TYPE
  (Value 110)........................................... C–41
IC_CFG_INVALID_TEMPLATE (Value 162) ... C–41
IC_CFG_INVALID_TYPE (Value 115)......... C–42
IC_CFG_INVALID_TYPE_SIZE
  (Value 135)........................................... C–42
IC_CFG_MISMATCH_DATA (Value 126)....... C–42
IC_CFG_NAME_TRUNCATED (Value 130)...... C–42
IC_CFG_NEW_DATA (Value 128).............. C–43
IC_CFG_NO_DATA_MEMORY (Value 136)...... C–43
IC_CFG_NO_HCFG_MEMORY
  (Value 139)........................................... C–43
IC_CFG_NO_HDB_MEMORY (Value 138)...... C–44
IC_CFG_NO_HLIB_MEMORY (Value 162)..... C–44
IC_CFG_NO_INFO_MEMORY (Value 137)..... C–44
IC_CFG_NO_INIT (Value 102)................. C–45
IC_CFG_NOT_FOUND (Value 125)............. C–45
IC_CFG_NOT_IMPLEMENTED
  (Value 101)........................................... C–45
IC_CFG_STILL_ACTIVE (Value 142)....... C–46
IC_CFG_UNKNOWN_COMPONENT
  (Value 119)........................................... C–46
IC_CFG_UNKNOWN_FIELDTYPE
  (Value 122)........................................... C–46
IC_CFG_UNKNOWN_GENERIC
  (Value 121)........................................... C–46
IC_CFG_UNKNOWN_PROPERTY
  (Value 118)........................................... C–47
IC_CFG_UNKNOWN_ROLE (Value 117)...... C–47
IC_CFG_UNKNOWN_SUPPLIER
  (Value 120)........................................... C–47
IC_CFG_UNSAVED_DATA (Value 129)...... C–47
## Contents

IC_CFG_WRONG_FIELDSIZE (Value 123) ... C–48
IC_CFG_WRONG_FIELDTYPE
  (Value 124) ............................................. C–48

IcACOMS........................................................................ C–49
IcACOMS Errors......................................................... C–49
  COMS_CHANNEL_ACTIVE (Value 225) ........ C–49
  COMS_ERROR_ACTIVESESS (Value 211) .... C–49
  COMS_ERROR_DUPLICATE (Value 214) .... C–50
  COMS_ERROR_INSERTCHANNEL
    (Value 216) ............................................. C–50
  COMS_ERROR_INSERTSESSION
    (Value 215) ............................................. C–50
  COMS_ERROR_INSERTWINDOWS
    (Value 217) ............................................. C–51
  COMS_ERROR_MAXDIALOGS
    (Value 212) ............................................. C–51

IcHLCNTS........................................................................ C–52
IcHLCNTS Errors...................................................... C–52
  NTS_CONNECT_DENIED (Value 3) .......... C–52
  NTS_CONNECT_FAILED (Value 2) ......... C–53
  NTS_CONNECT_LOST (Value 4) ............ C–53
  NTS_CONNECT_REJECTED (Value 22) ..... C–54
  NTS_CREDITS_EXCEEDED (Value 23) .... C–54
  NTS_MSG_OK (Value 1) ....................... C–54
  NTS_NO_HOSTPATH (Value 24) ............. C–55
  NTS_TERMINAL_ACTIVE (Value 21) ....... C–55

IcLCW ........................................................................... C–56
IcLCW Errors.......................................................... C–56

IcLocal........................................................................... C–57
IcLocal Errors ......................................................... C–57

IcMon........................................................................... C–58
IcMon Errors .......................................................... C–58
  ICMON_ERR_KEYVALUE (Value 500) ...... C–58
  ICMON_ERR_NODUPEOPTIONS
    (Value 502) ............................................. C–58
  ICMON_ERR_RANGEVALUE (Value 501) ... C–59

IcNBIOS......................................................................... C–60
IcNBIOS Errors........................................................ C–60
  NETBIOS_DUP_NAME (Value 4) .......... C–60
  NETBIOS_ERR_ADATA (Value 8) ......... C–60
  NETBIOS_ERR_ADD_NAME (Value 5) ...... C–61
  NETBIOS_ERR_CALL (Value 7) .......... C–61
  NETBIOS_ERR_CONNECT (Value 9) ...... C–61
  NETBIOS_ERR_DELETE_NAME
    (Value 11) ............................................. C–62
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETBIOS_ERR_LISTEN (Value 6)</td>
<td>NETBIOS_ERR_LISTEN (Value 6)</td>
<td>C–62</td>
</tr>
<tr>
<td>NETBIOS_ERR_RECEIVE (Value 32)</td>
<td>NETBIOS_ERR_RECEIVE (Value 32)</td>
<td>C–62</td>
</tr>
<tr>
<td>NETBIOS_ERR_RECEIVING (Value 22)</td>
<td>NETBIOS_ERR_RECEIVING (Value 22)</td>
<td>C–63</td>
</tr>
<tr>
<td>NETBIOS_ERR_SEND (Value 33)</td>
<td>NETBIOS_ERR_SEND (Value 33)</td>
<td>C–63</td>
</tr>
<tr>
<td>NETBIOS_ERR_SENDING (Value 23)</td>
<td>NETBIOS_ERR_SENDING (Value 23)</td>
<td>C–63</td>
</tr>
<tr>
<td>NETBIOS_INTERNAL (Value 10)</td>
<td>NETBIOS_INTERNAL (Value 10)</td>
<td>C–64</td>
</tr>
<tr>
<td>NETBIOS_NOT_FOUND (Value 3)</td>
<td>NETBIOS_NOT_FOUND (Value 3)</td>
<td>C–64</td>
</tr>
<tr>
<td>NETBIOS_XMT_BUSY (Value 21)</td>
<td>NETBIOS_XMT_BUSY (Value 21)</td>
<td>C–64</td>
</tr>
<tr>
<td>NETBIOS_ERR_LISTEN (Value 6)</td>
<td>NETBIOS_ERR_LISTEN (Value 6)</td>
<td>C–62</td>
</tr>
<tr>
<td>NETBIOS_ERR_RECEIVE (Value 32)</td>
<td>NETBIOS_ERR_RECEIVE (Value 32)</td>
<td>C–62</td>
</tr>
<tr>
<td>NETBIOS_ERR_RECEIVING (Value 22)</td>
<td>NETBIOS_ERR_RECEIVING (Value 22)</td>
<td>C–63</td>
</tr>
<tr>
<td>NETBIOS_ERR_SEND (Value 33)</td>
<td>NETBIOS_ERR_SEND (Value 33)</td>
<td>C–63</td>
</tr>
<tr>
<td>NETBIOS_ERR_SENDING (Value 23)</td>
<td>NETBIOS_ERR_SENDING (Value 23)</td>
<td>C–63</td>
</tr>
<tr>
<td>NETBIOS_INTERNAL (Value 10)</td>
<td>NETBIOS_INTERNAL (Value 10)</td>
<td>C–64</td>
</tr>
<tr>
<td>NETBIOS_NOT_FOUND (Value 3)</td>
<td>NETBIOS_NOT_FOUND (Value 3)</td>
<td>C–64</td>
</tr>
<tr>
<td>NETBIOS_XMT_BUSY (Value 21)</td>
<td>NETBIOS_XMT_BUSY (Value 21)</td>
<td>C–64</td>
</tr>
<tr>
<td>IcTCP</td>
<td>IcTCP Errors</td>
<td>C–65</td>
</tr>
<tr>
<td>IcTELNET</td>
<td>IcTELNET Errors</td>
<td>C–66</td>
</tr>
<tr>
<td>IcTELNET Errors</td>
<td>IcTELNET Errors</td>
<td>C–66</td>
</tr>
<tr>
<td>TELNET_BAD_CONFIG (Value 12)</td>
<td>TELNET_BAD_CONFIG (Value 12)</td>
<td>C–66</td>
</tr>
<tr>
<td>TELNET_ERR_RECEIVING (Value 22)</td>
<td>TELNET_ERR_RECEIVING (Value 22)</td>
<td>C–66</td>
</tr>
<tr>
<td>TELNET_ERR_SENDING (Value 23)</td>
<td>TELNET_ERR_SENDING (Value 23)</td>
<td>C–67</td>
</tr>
<tr>
<td>TELNET_INTERNAL (Value 10)</td>
<td>TELNET_INTERNAL (Value 10)</td>
<td>C–67</td>
</tr>
<tr>
<td>IcTrace</td>
<td>IcTrace Errors</td>
<td>C–68</td>
</tr>
<tr>
<td>IcTrace Errors</td>
<td>IcTrace Errors</td>
<td>C–68</td>
</tr>
<tr>
<td>IcTTY</td>
<td>IcTTY Errors</td>
<td>C–69</td>
</tr>
<tr>
<td>IcTTY Errors</td>
<td>IcTTY Errors</td>
<td>C–69</td>
</tr>
<tr>
<td>TTY_ERROR_BAUDERROR (Value 8)</td>
<td>TTY_ERROR_BAUDERROR (Value 8)</td>
<td>C–69</td>
</tr>
<tr>
<td>TTY_ERROR_BYTEERROR (Value 7)</td>
<td>TTY_ERROR_BYTEERROR (Value 7)</td>
<td>C–69</td>
</tr>
<tr>
<td>TTY_ERROR_DEFPARAM (Value 5)</td>
<td>TTY_ERROR_DEFPARAM (Value 5)</td>
<td>C–70</td>
</tr>
<tr>
<td>TTY_ERROR_DIALABORTED (Value 11)</td>
<td>TTY_ERROR_DIALABORTED (Value 11)</td>
<td>C–70</td>
</tr>
<tr>
<td>TTY_ERROR_NOPORT (Value 1)</td>
<td>TTY_ERROR_NOPORT (Value 1)</td>
<td>C–70</td>
</tr>
<tr>
<td>TTY_ERROR_NOQs (Value 4)</td>
<td>TTY_ERROR_NOQs (Value 4)</td>
<td>C–70</td>
</tr>
<tr>
<td>TTY_ERROR_NOTIMER (Value 10)</td>
<td>TTY_ERROR_NOTIMER (Value 10)</td>
<td>C–71</td>
</tr>
<tr>
<td>TTY_ERROR_NOTOPEN (Value 3)</td>
<td>TTY_ERROR_NOTOPEN (Value 3)</td>
<td>C–71</td>
</tr>
<tr>
<td>TTY_ERROR_OPEN (Value 2)</td>
<td>TTY_ERROR_OPEN (Value 2)</td>
<td>C–71</td>
</tr>
<tr>
<td>TTY_ERROR_UNAVAILPORT (Value 6)</td>
<td>TTY_ERROR_UNAVAILPORT (Value 6)</td>
<td>C–72</td>
</tr>
<tr>
<td>TTY_ERROR_UNKNOWN (Value 9)</td>
<td>TTY_ERROR_UNKNOWN (Value 9)</td>
<td>C–72</td>
</tr>
<tr>
<td>TTY_LCLERROR_FAILED (Value 40)</td>
<td>TTY_LCLERROR_FAILED (Value 40)</td>
<td>C–72</td>
</tr>
<tr>
<td>TTY_RCERROR_FAILED (Value 22)</td>
<td>TTY_RCERROR_FAILED (Value 22)</td>
<td>C–72</td>
</tr>
<tr>
<td>TTY_RCERROR_FRAME (Value 21)</td>
<td>TTY_RCERROR_FRAME (Value 21)</td>
<td>C–73</td>
</tr>
<tr>
<td>TTY_RCERROR_OVERRUN (Value 20)</td>
<td>TTY_RCERROR_OVERRUN (Value 20)</td>
<td>C–73</td>
</tr>
<tr>
<td>TTY_XMERROR_CTSTO (Value 30)</td>
<td>TTY_XMERROR_CTSTO (Value 30)</td>
<td>C–73</td>
</tr>
<tr>
<td>TTY_XMERROR_DSRTS (Value 31)</td>
<td>TTY_XMERROR_DSRTS (Value 31)</td>
<td>C–73</td>
</tr>
<tr>
<td>TTY_XMERROR_RLSDT (Value 32)</td>
<td>TTY_XMERROR_RLSDT (Value 32)</td>
<td>C–74</td>
</tr>
<tr>
<td>TTY_XMERROR_TRANSMITTING</td>
<td>TTY_XMERROR_TRANSMITTING</td>
<td>C–74</td>
</tr>
<tr>
<td>TTY_XMERROR_TXFULL (Value 33)</td>
<td>TTY_XMERROR_TXFULL (Value 33)</td>
<td>C–74</td>
</tr>
</tbody>
</table>
Contents

IcXNS ........................................................................................................ C–75
IcXNS Errors .................................................................................... C–75
   DCDEV_BAD_DEVICE (Value 1003) ........................................ C–75
   DCDEV_NO_CHANNEL (Value 1006) .......... C–75
   DCDEV_NO_DEVICE (Value 1001) ................ C–76
   DCDEV_NO_DRIVER (Value 1005) ................ C–76
   DCDEV_NOT_DEVICE (Value 1002) ............ C–76
   DCDEV_OLD_DEVICE (Value 1004) ............ C–77
   DCDEV_READ_ERROR (Value 1020) ............. C–77
   DCDEV_WRITE_ERROR (Value 1021) ............. C–77
   DCDEV_WRITE_INCOMPLETE
      (Value 1022) .................................................................. C–78
   XNS_ADDRESS_ERROR (Value 701) ............... C–78
   XNS_SOCKET_ERROR (Value 702) ................ C–78

Glossary ............................................................................................. G-1

Index ................................................................................................... I-1
About This Manual

Purpose

This reference manual, relative to release 3.0, provides detailed information about the INFOConnect Connectivity Services (ICS) programming interface, messages, and data types available for ICS Accessory development and for the development of additional data filters (Service Libraries) and connection types (External Interface Libraries).

Scope

This is a Basic INFOConnect Developer's Kit. This manual is intended purely as a reference for use in developing components to the INFOConnect Connectivity Services product.

Audience

The INFOConnect Development Kit Basic Programming Reference Manual audience is the programmer who is developing cooperative applications that use INFOConnect Connectivity Services for data communications, or developing reusable INFOConnect accessories. This manual is also geared towards the developer who wishes to build additional data filters (Service Libraries) and connection types (External Interface Libraries). For information on the concepts and procedures involved in developing ICS components, refer to the INFOConnect Development Kit Basic Developer's Guide.
Prerequisites

Applications dependent on Microsoft® Windows™ 3.0 or 3.1 (referred to in this document as Windows or MS-Windows) must be familiar with the Windows Software Development Kit. Familiarity with a C language compiler compatible with Microsoft Windows 3.0 or 3.1 is also necessary.

XVT™ is the presentation toolkit that is supported by Unisys for developing portable applications on the MS-Windows platform. Therefore, the programmer wishing to develop portable user interface code using XVT must be familiar with the XVT Presentation Toolkit.

How to Use This Guide

This is a reference manual. It is meant to be used as a reference tool in conjunction with the INFOConnect Development Kit Basic Developer’s Guide.

Organization

This manual consists of the following sections and appendixes. In addition, a glossary and an index appear at the end of this manual.

Section 1. Introduction

This section provides background information about the INFOConnect Connectivity Services program and, in particular, about the INFOConnect Development Kit.

Section 2. Functions by Category

This section lists and briefly describes the ICS API functions according to these categories:

- Accessory API
- DosLink API
- Memory Management API
- General Utilities
- Library API

Section 3. INFOConnect Connectivity Services API

This section contains an alphabetical list of the ICS API. The documentation for each function includes the function prototype, a description of the function, an explanation of each of the parameters, and the possible return values. Also included are any special notes about use of the function, as well as a key table noting which ICS layer would use the function. Related topics, such as specific data types or events/messages related to the API, are also listed.
About This Manual

Section 4. ICS Messages/Events
This section contains an alphabetical list and documentation for the Windows messages and XVT/Win events defined by INFOConnect Connectivity Services.

Section 5. ICS Data Structures/Types
This section contains an alphabetical list and documentation for the data structures and types defined by INFOConnect Connectivity Services.

Section 6. ICS Accessory Definition
This section describes the components of an ICS Accessory.

Appendix A. Standard IDs (Keys) & Component Numbers
Appendix A lists and describes the INFOConnect Connectivity Services standard IDs for accessories and libraries, as well as the standard component numbers and currently assigned vendor-specific component numbers.

Appendix B. Status Types and Statuses
Appendix B lists and describes the INFOConnect Connectivity Services status types and statuses.

Appendix C. Errors and Results
Appendix C lists and describes the INFOConnect Connectivity Services errors and informative results, as well as errors specific to Unisys-provided ICS components.
About This Manual

Related Product Information

INFOConnect™ Development Kit Basic Developer's Guide
(4173 5408-000)
Describes how to use the IDK to develop INFOConnect Accessories, and to develop additional data filters (Service Libraries) and connection types (External Interface Libraries).

INFOConnect™ Connectivity Services Installation and Configuration Guide (4240 0119-200)
Contains information on installing and configuring the INFOConnect runtime product.

Microsoft® Windows™ Software Development Kit Programmer's Reference
Contains reference material for the Windows SDK.

Microsoft® Windows™ Software Development Kit Guide to Programming
Describes how to use the Windows SDK to develop Windows applications and dynamic link libraries.

XVT™ Programmer's Manual
Contains reference material for the XVT developers kit.
## Notational Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessory</td>
<td>When selected in the key table of a specific INFOConnect API description, indicates that this API is specific to INFOConnect Accessories through the Accessory AIL. Note that the API is available to those INFOConnect applications and libraries that have initialized themselves by calling the <code>IcInitics</code> function.</td>
</tr>
<tr>
<td>AIL</td>
<td>Abbreviation for Application Interface Library. When selected in the key table of a specific INFOConnect API description, indicates that this API must either be provided by the AIL or is available as a utility to the AIL. Note that functions that begin with <code>IcLib...</code> must be provided by the AIL at the given ordinal values. Unless otherwise stated, the term AIL also refers to its various forms, such as interprocess interface library and stack interface library.</td>
</tr>
<tr>
<td>Bold</td>
<td>Function names and data types/structures appear in bold.</td>
</tr>
<tr>
<td>Byte</td>
<td>In this document, this term is equivalent to one octet.</td>
</tr>
<tr>
<td>Configurator</td>
<td>When selected in the key table of a specific INFOConnect API description, indicates that this API is specific to Configuration Accessories. The API is available to those INFOConnect accessories and libraries that have initialized themselves as a Configurator by calling the <code>IcOpenDatabase</code> function.</td>
</tr>
</tbody>
</table>
## About This Manual

### Convention Description

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DosLink</td>
<td>When selected in the key table of a specific INFOConnect API description, indicates that this API is specific to those DOS applications that run in MS-Windows Enhanced Mode and use INFOConnect Connectivity Services for data communications.</td>
</tr>
<tr>
<td>EIL</td>
<td>Abbreviation for External Interface Library. When selected in the key table of a specific INFOConnect API description, indicates that this API must either be a callback function provided by the EIL or is available as a utility to the EIL. Note that functions that begin with <strong>IcLib</strong>... must be provided by the EIL at the given ordinal values.</td>
</tr>
<tr>
<td>HI</td>
<td>Refers to the high word (high-order 16 bits) of a long parameter.</td>
</tr>
<tr>
<td>IN, OUT</td>
<td>In parameter description, indicates if the parameter is input (IN), output (OUT), or both input/output (IN/OUT).</td>
</tr>
<tr>
<td>*IN, *OUT</td>
<td>In parameter description of pointer parameters, indicates whether the data POINTED to is input (*IN), output (*OUT), or both input/output (*IN/*OUT). The pointer parameter itself must always be input.</td>
</tr>
<tr>
<td>italicized words</td>
<td>Function parameters and fields of a data structure are italicized.</td>
</tr>
<tr>
<td>LO</td>
<td>Refers to the low word (low-order 16 bits) of a long parameter.</td>
</tr>
</tbody>
</table>
### About This Manual

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>In parameter description, indicates that the value of the input parameter is not defined. Use the appropriate NULL value.</td>
</tr>
<tr>
<td>Shell</td>
<td>When selected in the key table of a specific INFOConnect API description, indicates that this API is specific to INFOConnect Shells. Note that the API is also available to those INFOConnect accessories or libraries that have initialized themselves as a Shell by calling the <code>IcInitShell</code> function.</td>
</tr>
<tr>
<td>SL</td>
<td>Abbreviation for Service Library. When selected in the key table of a specific INFOConnect API description, indicates that this API must either be provided by the SL or is available as a utility to the SL. Note that functions that begin with <code>IcLib...</code> must be provided by the SL at the given ordinal values.</td>
</tr>
<tr>
<td>(ver)</td>
<td>In the function heading, indicates the first version of the INFOConnect Development Kit in which the given API is available.</td>
</tr>
<tr>
<td>WIN</td>
<td>Abbreviation for Windows. When selected in the key table of a specific INFOConnect API description, indicates that this API is Windows-specific.</td>
</tr>
<tr>
<td>XVT</td>
<td>When selected in the key table of a specific INFOConnect API description, indicates that this API is XVT/Win-specific.</td>
</tr>
</tbody>
</table>
INFOConnect Connectivity Services provides a Microsoft Windows version of the communications interface, as well as an interface utilizing XVT for Windows. The ICS function names follow the style conventions of the Windows platform. XVT/Win-specific API follows the style conventions of that platform.

Function names are constructed by using the IC prefix followed by a verb/noun combination. This combination indicates the action (such as the verb open) of the function and the object (such as the noun session) on which the function operates. Each word in the function name begins with a capital letter (for example, **IcOpenSession**). The XVT/Win-specific API function names are in lower case with each word in the name separated by an underscore (for example, **ic_open_session**).

All #defined names are capitalized (for example, **IC_RCVDONE**).

ICS events for the Windows platform must be registered with Windows. This is done using the quoted version described in the Events/Messages section of this manual (for example, "**IC_RcvDone**"). ICS events for the XVT/Win platform are in all capital letters and are prefixed by **E_IC_** (for example, **E_IC_RCV_DONE**).
Section 1
Introduction

The INFOConnect Connectivity Services (ICS) Program provides a workstation platform that delivers code portability and reusability to the developer of a cooperative system. ICS provides an open, layered architecture that allows application independence from session/presentation-type services and from specific data communications protocols. This is achieved by addressing many known limitations and differences among Graphical User Interfaces (GUIs), communication protocols and other aspects of supporting cooperative systems.

The INFOConnect Development Kit (IDK) provides the tools required for a developer to build portable and reusable components for the INFOConnect product. By using the IDK, developers can create ICS components that can plug into the various layers of the ICS architecture. This IDK consists of the INFOConnect Development Kit Basic Programming Reference Manual (this document), the INFOConnect Development Kit Basic Developer’s Guide, and the INFOConnect Developer’s Diskette(s). The IDK Diskette(s) contain libraries that provide a consistent application programming interface (API) across the various platforms supported, as well as many sample INFOConnect components. Developers who utilize the INFOConnect Development Kit can be assured that all components documented within this Kit will work together.
Introduction

ICS Accessory API

The INFOConnect Connectivity Services Accessories API provides both a Microsoft Windows version of the INFOConnect Connectivity Services API and a platform independent version utilizing XVT/Win. XVT/Win-specific API is provided for those ICS functions that require a buffer handle parameter. Use the ICS provided memory utilities for XVT/Win to obtain global buffers. To access ICS functions that have a Windows window handle parameter, use GET_HWND() under XVT 2.0, or first use the XVT/Win get_value() function to obtain the Windows ATTR_NATIVE_WINDOW window handle under XVT 3.

INFOConnect also provides an API for DOS applications that run under MS-Windows Enhanced mode and wish to use the MS-Windows version of the INFOConnect API for client/server-type data communications. See the INFOConnect Development Kit Basic Developer's Guide for information on writing these types of applications.

ICS Library API

Since XVT does not currently support the development of dynamic link libraries, service libraries and external interface libraries must be developed for specific platforms. Therefore, INFOConnect Connectivity Services provides a GUI specific programming interface for developing these libraries. Each library is required to provide a core set of functions and may utilize the ICS programming interface. See Section 2, "ICS Library API" for a brief description of the ICS programming interface for library development.
Section 2
Functions By Category

This section lists and briefly describes the ICS API functions according to these categories:

- Accessory API
- DosLink API
- Memory Management API
- General Utilities
- Library API

ICS Accessory API

The ICS Accessory Application Programming Interface (AAPI) is the interface provided for accessories and applications used in session management and error handling. The INFOConnect Accessory AIL (IcAAPI16.DLL) exports this AAPI. Note that the AAPIs exist in the ICS Manager and are, by default, available to the INFOConnect accessory. Both MS-Windows and XVT/Win versions of the AAPI are provided.
**MS-Windows API**

To access Windows-specific AAPI, messages, and data types, include the `icwin.h` include file after `WINDOWS.H`.

### Basic Session Management Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcCloseSession</td>
<td>Initiates session termination.</td>
</tr>
<tr>
<td>IcExitOk</td>
<td>Responds to an ICS exit request.</td>
</tr>
<tr>
<td>IcInitIcs</td>
<td>Initializes ICS.</td>
</tr>
<tr>
<td>IcLcl</td>
<td>Cancels pending transmits and/or receives.</td>
</tr>
<tr>
<td>IcOpenAccessory</td>
<td>Starts an ICS accessory with a local connection.</td>
</tr>
<tr>
<td>IcOpenSession</td>
<td>Initiates session establishment.</td>
</tr>
<tr>
<td>IcRcv</td>
<td>Requests a buffer of data.</td>
</tr>
<tr>
<td>IcRegisterMsgSession</td>
<td>Registers ICS events.</td>
</tr>
<tr>
<td>IcXmt</td>
<td>Initiates transmission of a data buffer.</td>
</tr>
</tbody>
</table>

### Additional Session Management Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcChangeHandle</td>
<td>Changes the ownership of an open session.</td>
</tr>
<tr>
<td>IcGetPathID</td>
<td>Obtains the path ID of an active session.</td>
</tr>
<tr>
<td>IcGetSessionID</td>
<td>Obtains a session identification string from a session handle.</td>
</tr>
<tr>
<td>IcGetSessionInfo</td>
<td>Returns pertinent information about a session.</td>
</tr>
<tr>
<td>IcSetStatus</td>
<td>Sends a status message.</td>
</tr>
</tbody>
</table>
Functions By Category

Error Handling

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcDefaultErrorProc</td>
<td>Allows ICS to handle an error result.</td>
</tr>
<tr>
<td>IcGetString</td>
<td>Converts an error result into a string.</td>
</tr>
<tr>
<td>IcSetError</td>
<td>Used by accessories to generate errors.</td>
</tr>
</tbody>
</table>

XVT/Win API

The XVT/Win API, events, and data types are made available to your application by INFOConnect Connectivity Services through the **XVT.H** include file. This is done by running the ICXVTMOD utility (See the **IDK Basic Developer's Guide** for more information on installing the IDK). Therefore, there is no additional file to include in order to access these functions.

Basic Session Management Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_close_session</td>
<td>Initiates session termination.</td>
</tr>
<tr>
<td>ic_exit_ok</td>
<td>Responds to an ICS exit request.</td>
</tr>
<tr>
<td>ic_init_ics</td>
<td>Initializes ICS.</td>
</tr>
<tr>
<td>ic_lcl</td>
<td>Cancels pending transmits and/or receives.</td>
</tr>
<tr>
<td>ic_open_accessory</td>
<td>Starts an ICS accessory with a local connection.</td>
</tr>
<tr>
<td>ic_open_session</td>
<td>Initiates session establishment.</td>
</tr>
<tr>
<td>ic_rcv</td>
<td>Requests a buffer of data.</td>
</tr>
<tr>
<td>ic_register_msg_session</td>
<td>Registers ICS events.</td>
</tr>
<tr>
<td>ic_xmt</td>
<td>Initiates transmission of a data buffer.</td>
</tr>
</tbody>
</table>
### Functions By Category

#### Additional Session Management Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_change_handle</td>
<td>Changes the ownership of an open session.</td>
</tr>
<tr>
<td>ic_get_path_id</td>
<td>Obtains the path identification string of an active session.</td>
</tr>
<tr>
<td>ic_get_session_id</td>
<td>Obtains a session identification string from a session handle.</td>
</tr>
<tr>
<td>ic_get_session_info</td>
<td>Returns pertinent information about a session.</td>
</tr>
<tr>
<td>ic_set_status</td>
<td>Sends a status message.</td>
</tr>
</tbody>
</table>

#### Error Handling

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_default_error_proc</td>
<td>Allows ICS to handle an error result.</td>
</tr>
<tr>
<td>ic_get_string</td>
<td>Converts an error result into a string.</td>
</tr>
<tr>
<td>ic_set_error</td>
<td>Used by accessories to generate errors.</td>
</tr>
</tbody>
</table>
ICS DosLink Client/Server Applications

DosLink Client/Server-type applications may access the MS-Windows version of the Basic Session Management Functions, Additional Session Management Functions, Memory Management Functions and the **IcSetError** function, as well as the functions listed below. To access this API, and the ICS messages and data types, include the *icdos.h* include file. DosLink Client/Server applications can run in Windows enhanced mode only, and DosLinkS.EXE must be running. See the *IDK Basic Developer's Guide* for more information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcCreateHandle</td>
<td>Creates an ICS memory handle from a DOS far string pointer with offset zero.</td>
</tr>
<tr>
<td>IcCreateHwnd</td>
<td>Creates an ICS window handle.</td>
</tr>
<tr>
<td>IcCreateSession</td>
<td>Creates an ICS session structure.</td>
</tr>
<tr>
<td>IcDestroyHandle</td>
<td>Destroys the handle created by IcCreateHandle.</td>
</tr>
<tr>
<td>IcDestroyHwnd</td>
<td>Destroys the handle created by IcCreateHwnd.</td>
</tr>
<tr>
<td>IcDestroySession</td>
<td>Destroys an ICS session structure.</td>
</tr>
<tr>
<td>IcGetNextEvent</td>
<td>Retrieves the next event for a session.</td>
</tr>
<tr>
<td>IcGetServiceName</td>
<td>Retrieves the service name of the partner session.</td>
</tr>
<tr>
<td>IcHandleOffset</td>
<td>Sets the memory offset for the DOS far string pointer.</td>
</tr>
<tr>
<td>IcNextEvent</td>
<td>Indicates the callback routine is ready for the next event.</td>
</tr>
<tr>
<td>IcRegisterCallback</td>
<td>Registers a session's callback routine.</td>
</tr>
<tr>
<td>IcSetServerInfo</td>
<td>Declares a session to be a server session.</td>
</tr>
</tbody>
</table>
ICS Memory Management API

MS-Windows API

This memory management API is accessible by ICS accessories and libraries. No additional include file is needed in order to access this API.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcAllocBuffer</td>
<td>Allocates sharable memory.</td>
</tr>
<tr>
<td>IcFreeBuffer</td>
<td>Frees memory allocated with IcAllocBuffer.</td>
</tr>
<tr>
<td>IcGetBufferSize</td>
<td>Returns the size of a buffer allocated with IcAllocBuffer.</td>
</tr>
<tr>
<td>IcLockBuffer</td>
<td>Locks memory allocated with IcAllocBuffer.</td>
</tr>
<tr>
<td>IcReadBuffer</td>
<td>Reads data from a buffer.</td>
</tr>
<tr>
<td>IcReAllocBuffer</td>
<td>Resizes memory allocated with IcAllocBuffer.</td>
</tr>
<tr>
<td>IcUnlockBuffer</td>
<td>Unlocks memory locked with IcLockBuffer.</td>
</tr>
<tr>
<td>IcWriteBuffer</td>
<td>Writes data to a buffer.</td>
</tr>
</tbody>
</table>

XVT/Win API

This API is defined in XVT.H. No additional include file is needed to access this API.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_buf_alloc</td>
<td>Allocates sharable memory.</td>
</tr>
<tr>
<td>ic_buf_free</td>
<td>Frees memory allocated with ic_buf_alloc.</td>
</tr>
<tr>
<td>ic_buf_lock</td>
<td>Locks memory allocated with ic_buf_alloc.</td>
</tr>
<tr>
<td>ic_buf_realloc</td>
<td>Resizes memory allocated with ic_buf_alloc.</td>
</tr>
</tbody>
</table>
### Functions By Category

- **IC_CHECK_DATAFLAGS**
  - Unlocks memory locked with `ic_buf_lock`
- **IC_GALLOC**
  - Allocates non-sharable memory.
- **IC_GFREE**
  - Frees memory allocated with `ic_galloc`.
- **IC_GLLOCK**
  - Locks memory allocated with `ic_galloc`.
- **ICGREALLOC**
  - Resizes memory allocated with `ic_galloc`.
- **IC_GUNLOCK**
  - Unlocks memory locked with `ic_glock`.

### General Utilities

These general utilities are accessible by ICS accessories and libraries. No additional include file is needed in order to access these general utilities under either platform.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_CHECK_DATAFLAGS</td>
<td>Use to retrieve the value of an <code>IC_STATUS_DATAFLAGS</code> status.</td>
</tr>
<tr>
<td>IC_CHECK_RESULT_SEVERE</td>
<td>Use to check the severity of an <code>IC_RESULT</code></td>
</tr>
<tr>
<td>IC_GET_RESULT_CONTEXT</td>
<td>Use to extract the context from an <code>IC_RESULT</code></td>
</tr>
<tr>
<td>IC_GET_RESULT_SUBTYPE</td>
<td>Use to extract the subtype from an <code>IC_RESULT</code></td>
</tr>
<tr>
<td>IC_GET_RESULT_SUBVALUE</td>
<td>Use to extract the subvalue from an <code>IC_RESULT</code></td>
</tr>
<tr>
<td>IC_GET_RESULT_TYPE</td>
<td>Use to extract the type from an <code>IC_RESULT</code></td>
</tr>
<tr>
<td>IC_GET_RESULT_VALUE</td>
<td>Use to extract the value from an <code>IC_RESULT</code></td>
</tr>
<tr>
<td>IC_MAKE_RESULT</td>
<td>Creates an <code>IC_RESULT</code> from a context, a type, and a value.</td>
</tr>
<tr>
<td>IcRunHelp3</td>
<td>Invokes the ICS help system.</td>
</tr>
</tbody>
</table>
## MS-Windows API

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcGetINFOConnectDir</td>
<td>Use to obtain CodeDir or DataDir.</td>
</tr>
<tr>
<td>IcMgrTraceBuffer</td>
<td>Writes a buffer of data to IcTrace's debug file.</td>
</tr>
<tr>
<td>IcMgrTraceResult</td>
<td>Writes an <strong>IC_RESULT</strong> to IcTrace's debug file.</td>
</tr>
</tbody>
</table>

### Path Management Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcGetNewPath</td>
<td>Initiates a path configuration dialog.</td>
</tr>
<tr>
<td>IcGetPathNames</td>
<td>Provides a list of configured paths.</td>
</tr>
</tbody>
</table>

### Accessory Management

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcDeregisterAccessory</td>
<td>Companion to <strong>IcRegisterAccessory</strong>.</td>
</tr>
<tr>
<td>IcGetContext</td>
<td>Converts a string identifier into a context.</td>
</tr>
<tr>
<td>IcGetContextString</td>
<td>Converts a context into a string identifier.</td>
</tr>
<tr>
<td>IcRegisterAccessory</td>
<td>Identifies the application as an accessory.</td>
</tr>
<tr>
<td>IcRunAccessory</td>
<td>Independently starts an ICS accessory.</td>
</tr>
</tbody>
</table>

### Accessory-Only Utilities

The following general utilities are available for ICS accessories only. Include the `icutil.h` include file to access them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcGetCmdlineOption</td>
<td>Retrieves a given command line option.</td>
</tr>
</tbody>
</table>
### Functions By Category

#### XVT/Win API

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_get_infoconnect_dir</td>
<td>Use to obtain CodeDir or DataDir.</td>
</tr>
</tbody>
</table>

#### Path Management Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_get_new_path</td>
<td>Initiates a path configuration dialog.</td>
</tr>
<tr>
<td>ic_get_path_names</td>
<td>Provides a list of configured paths.</td>
</tr>
</tbody>
</table>

#### Accessory Management

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_deregister_accessory</td>
<td>Companion to <code>ic_register_accessory</code>.</td>
</tr>
<tr>
<td>ic_get_context</td>
<td>Converts a string identifier into a context.</td>
</tr>
<tr>
<td>ic_get_context_string</td>
<td>Converts a context into a string identifier.</td>
</tr>
<tr>
<td>ic_register_accessory</td>
<td>Identifies the application as an accessory.</td>
</tr>
<tr>
<td>ic_run_accessory</td>
<td>Independently starts an ICS accessory.</td>
</tr>
</tbody>
</table>
ICS Library API

Entry Points Provided by SLs and EILs

Each of the following procedures must be exported by SLs and EILs at the given ordinal numbers. To view the prototypes associated with these IcLib... procedures, see the icproto.h include file. The library should use procedure names that more closely adhere to its purpose. For example, a TTY EIL could use function names that begin with IcTTY..., and a COMS SL could use names that begin with IcCOMS... The specific library's .DEF file references the function names used by that implementation.

The library procedures listed under Library Load/Unload, Session Establishment, and Session Communications are guaranteed to be called under the INFOConnect Shell's task.

The icdict.h file is included into the library's resource file to support the ICS required resources. See the ICS Data Structures/Types section for information on the structure of these user-defined resources. Refer to Microsoft® Windows™ Software Development Kit, Programmer's Reference, User-Defined Resource Statement section for more information.

Since XVT does not currently support the development of dynamic link libraries, service libraries and external interface libraries must be developed for specific platforms.
## MS-Windows API

### Library Load/Unload

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcLibInstall @ 6</td>
<td>First procedure called by ICS when the library is loaded.</td>
</tr>
<tr>
<td>IcLibTerminate @ 12</td>
<td>Last procedure called by ICS before the library is unloaded.</td>
</tr>
</tbody>
</table>

### Session Establishment

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcLibOpenChannel @ 16</td>
<td>Called when the channel is first opened.</td>
</tr>
<tr>
<td>IcLibCloseChannel @ 17</td>
<td>Procedure called when the channel is no longer needed.</td>
</tr>
<tr>
<td>IcLibOpenSession @ 9</td>
<td>Called to open a session on the given channel.</td>
</tr>
<tr>
<td>IcLibCloseSession @ 2</td>
<td>Procedure called to close the session.</td>
</tr>
<tr>
<td>IcLibIdentifySession @ 5</td>
<td>Called to uniquely identify a session.</td>
</tr>
</tbody>
</table>

### Session Communications

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcLibEvent @ 3</td>
<td>Called to process ICS messages.</td>
</tr>
<tr>
<td>IcLibXmt @ 13</td>
<td>Procedure called to transmit data.</td>
</tr>
<tr>
<td>IcLibRecv @ 8</td>
<td>Procedure called to receive data.</td>
</tr>
<tr>
<td>IcLibLcl @ 7</td>
<td>Called to cancel pending transmits and/or receives.</td>
</tr>
<tr>
<td>IcLibSetResult @ 11</td>
<td>Called to process status and error messages.</td>
</tr>
</tbody>
</table>
### Functions By Category

#### Session Information

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcLibGetSessionInfo @ 10</td>
<td>Procedure called to provide session related information.</td>
</tr>
<tr>
<td>IcLibGetString @ 4</td>
<td>Called to convert an error result into a string.</td>
</tr>
</tbody>
</table>

#### Session Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcLibUpdateConfig @ 1</td>
<td>Procedure called to update configuration.</td>
</tr>
<tr>
<td>IcLibVerifyConfig @ 14</td>
<td>Called to verify the contents of a configuration buffer.</td>
</tr>
<tr>
<td>IcLibPrintConfig @ 15</td>
<td>Procedure called to obtain displayable configuration information.</td>
</tr>
</tbody>
</table>

#### ICS Utilities for Library Development

To access the library utilities API, messages, and data types, include the `iclib.h` include file after `WINDOWS.H`.

#### MS-Windows API

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcAddRefContextID</td>
<td>Locks a library into memory until it is released.</td>
</tr>
<tr>
<td>IcGetChannelID</td>
<td>Obtains the ID of a channel.</td>
</tr>
<tr>
<td>IcGetContextID</td>
<td>Obtains the context of a library and locks the library until it is released. Loads the library if not already loaded.</td>
</tr>
<tr>
<td>IcIsDebug</td>
<td>Obtains the current debug mode of ICS.</td>
</tr>
<tr>
<td>IcMgrEilEvent</td>
<td>Posts events to an EIL's event procedure.</td>
</tr>
<tr>
<td>IcMgrGetSessionInfo</td>
<td>Returns pertinent information about the lower a part of a session.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IcMgrLcl</td>
<td>Sends local requests down the library stack.</td>
</tr>
<tr>
<td>IcMgrRcv</td>
<td>Sends receive requests down the library stack.</td>
</tr>
<tr>
<td>IcMgrSendEvent</td>
<td>Posts events upwards in the library stack.</td>
</tr>
<tr>
<td>IcMgrSetResult</td>
<td>Sends status and error results down the library stack.</td>
</tr>
<tr>
<td>IcMgrXmt</td>
<td>Sends transmit requests down the library stack.</td>
</tr>
<tr>
<td>IcNotifyConfig</td>
<td>Passes notification messages to the active configurators.</td>
</tr>
<tr>
<td>IcReleaseContextID</td>
<td>Releases the context of a library and decrements its reference count.</td>
</tr>
<tr>
<td></td>
<td>Companion to <strong>IcAddRefContextID</strong> and <strong>IcGetContextID</strong>.</td>
</tr>
<tr>
<td>IcRunLibHelp</td>
<td>Invokes the ICS help system.</td>
</tr>
<tr>
<td>IcSetSessionError</td>
<td>Records errors.</td>
</tr>
</tbody>
</table>
Section 3
INFOConnect API

This section fully documents, in alphabetical order, all of the INFOConnect API. Note that alphanumerics precede underscores.

The information given for each function includes the function prototype, a description of the function, an explanation of each of the function parameters, and the function's possible return values. Following this are any special notes about the function. Included is a table that flags the ICS component that would use the function. The first line of the table indicates the platform for which the function is geared: Windows, XVT/Win, or DOS (DosLink). The rest of the table indicates the ICS layer that would use the function: accessory, ICS Shell, ICS Configuration Accessories, Application Interface Library, Service Library, or External Interface Library. Following the table is a list of additional functions, data types, and messages/events that are related to the function and may, therefore, provide additional information.

This manual is part of the Basic INFOConnect Developer’s Kit.
IcAddRefContextID

IC_RESULT FAR PASCAL IcAddRefContextID
   ( IC_RESULT_CONTEXT context )

IcAddRefContextID is used to delay unloading the given library from memory. This may be necessary to ensure that the library remains in memory until after the library has completed the processing initiated by IcLibCloseChannel. The library is guaranteed to remain in memory until after a matching IcReleaseContextID is called.

Note that IcAddRefContextID increments the reference count for the given library. When use of the library is completed, IcReleaseContextID must be called to decrement the reference count. After the count reaches zero, the library's termination routine will be called and the library will be unloaded from memory.

Parameters

<table>
<thead>
<tr>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>IN An IC_RESULT_CONTEXT of the library to lock into memory.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. See Appendix C for possible errors.

Note: IcAddRefContextID and IcReleaseContextID must occur in matching pairs.

WIN XVT DosLink

Accessory Shell Configurator

AIL SL EIL

See also:

IcReleaseContextID function
IcAllocBuffer

HANDLE FAR PASCAL IcAllocBuffer
( unsigned bufsize )

IcAllocBuffer allocates a global buffer that can be shared by different tasks.

Parameters Description
bufsize IN The number of bytes to allocate.

Return Value:
A global buffer handle is returned if the memory was allocated. (HANDLE)NULL is returned if the memory could not be allocated.

Note: ICS data communication buffers must be shared by different tasks. IcAllocBuffer ensures that these buffers are properly allocated to satisfy any operating system requirements for shared buffers. Therefore, buffers passed to the INFOConnect Connectivity Services routines MUST have been allocated through IcAllocBuffer.

WIN  ❍ XVT  ● DosLink

● Accessory  ● Shell  ● Configurator
● AIL  ● SL  ● EIL

See also:
IcFreeBuffer function
IcChangeHandle

IC_RESULT FAR PASCAL IcChangeHandle
    ( HIC_SESSION hsession,
      HWND hWnd)

IcChangeHandle changes the ownership of a currently established communication session. All subsequent communication messages are then directed to the window function associated with that new window.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>The HIC_SESSION handle of the opened communication session to which the new window becomes associated.</td>
</tr>
<tr>
<td>hWnd</td>
<td>The window handle for the window that will obtain ownership of the given communication session.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the change was successful.
IC_ERROR_UNOPENEDSESSION is returned if the given communication session is not a valid, established session. See Appendix C for other possible errors.
Note: An implicit `IcLcl(hsession, IC_LCL_RCVXMT)` is performed prior to the switch.

- WIN
- Accessory
- AIL
- DosLink
- XVT
- Shell
- SL
- Configurator
- EIL

See also:
- `IcLcl` function
- `HIC_SESSION` data type
- `IC_LCL_FLAGS` data type
- `IcNextEvent` function
- `IC_NEXTEVENT_FLAGS` data type
IC_CHECK_DATAFLAGS

IC_CHECK_DATAFLAGS ( r )

The IC_CHECK_DATAFLAGS macro checks a status message to determine if it is an IC_STATUS_DATAFLAGS status.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>IN</td>
</tr>
</tbody>
</table>

The status result.

Return Value:

TRUE if the status is an IC_STATUS_DATAFLAGS status. FALSE otherwise.

● WIN  ● XVT  ○ DosLink

● Accessory  ○ Shell  ○ Configurator
● AIL  ● SL  ● EIL

See also:

IC_STATUS_DATAFLAGS data type
IC_CHECK_RESULT_SEVERE

IC_CHECK_RESULT_SEVERE ( result )

The IC_CHECK_RESULT_SEVERE macro checks the severity of the given IC_RESULT.

Parameters

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
</tr>
</tbody>
</table>

IN An IC_RESULT to check.

Return Value:

TRUE if the IC_RESULT_TYPE is IC_ERROR_SEVERE or IC_ERROR_TERMINATE. FALSE otherwise.

WIN ● XVT ● DosLink

● Accessory ● Shell ● Configurator

● AIL ● SL ● EIL

See also:

IC_ERROR_SEVERE data type
IC_ERROR_TERMINATE data type
IC_RESULT data type
IC_RESULT_TYPE data type
IcCloseSession

IC_RESULT FAR PASCAL IcCloseSession
    ( HIC_SESSION hsession )

IcCloseSession causes INFOConnect Connectivity Services to close the given communication session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned. The result of the communication session closure will be sent to the application's window procedure as the ICS message IC_SESSIONCLOSED. This result will be IC_OK if the session closed properly. See Appendix C for possible errors.

Notes:

− An IC_OK result from IcOpenSession requires that IcCloseSession be called regardless of the IC_SESSIONESTABLISHED message result.
− ICS DosLink applications should call IcDestroySession after calling this routine to flush the event buffer and destroy the session record.

See also:

IcOpenSession function
IcDestroySession function
IcNextEvent function
IC_NEXTEVENT_FLAGS data type
IcCreateHandle

HANDLE FAR PASCAL IcCreateHandle
    ( LPSTR mem,
        WORD len )

For ICS DosLink applications, IcCreateHandle creates a memory handle for use with the ICS API.

ICS DosLink applications would normally use IcAllocBuffer to obtain memory buffers. IcCreateHandle can be used instead only if the application's memory pointer has a zero offset. If the application's memory pointer does not have a zero offset, see the IcHandleOffset function.

Parameters Description

mem    IN    A pointer to the memory which the new handle should reference.

len    IN    The size of the data, in bytes.

Return Value:
The ICS memory handle is returned. NULL if the handle could not be created, that is, the pointer offset is not equal to zero.

❍ WIN    ❍ XVT    ● DosLink

● Accessory    ❍ Shell    ❍ Configurator
❍ AIL    ❍ SL    ❍ EIL

See also:
IcAllocBuffer function
IcDestroyHandle function
IcHandleOffset function
INFOConnect API

IcCreateHwnd

HWND FAR PASCAL IcCreateHwnd ( LPSTR classname )

For ICS DosLink applications, IcCreateHwnd creates an MS-Windows type window handle for use with the ICS API.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>classname</td>
<td>*IN An MS-Windows class, or, to use the ICS default, this could be a NULL string or the pointer itself may be NULL.</td>
</tr>
</tbody>
</table>

Return Value:

An ICS window handle is returned. NULL is returned if the handle could not be created.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL
IcCreateSession

IC_RESULT FAR PASCAL IcCreateSession
( LPHIC_SESSION lpsession )

For ICS DosLink applications, **IcCreateSession** creates an ICS session structure and returns its session handle. This handle must be passed in on the call to **IcOpenSession**.

If the ICS DosLink application is using a callback function (in contrast to polling using **IcGetNextEvent**), then **IcRegisterCallback** must be called before calling **IcOpenSession**. If the ICS DosLink application wishes to be a server session (instead of defaulting to a client session), then **IcSetServerInfo** must be called before calling **IcOpenSession**.

**Parameters**

**lpSession**

*OUT An **HIC_SESSION** to be initialized with the ICS session handle.

**Return Value**: 

**IC_OK** is returned if successful. See Appendix C for possible errors.

- **WIN**
- **XVT**
- **DosLink**

- **Accessory**
- **Shell**
- **Configurator**

- **AIL**
- **SL**
- **EIL**

**See also**:

- **IcOpenSession** function
- **IcRegisterCallback** function
- **IcSetServerInfo** function
- **IcDestroySession** function
IcDefaultErrorProc

IC_RESULT FAR PASCAL IcDefaultErrorProc
   ( HWND hWnd,
     HANDLE hData,
     unsigned uType,
     IC_RESULT error )

IcDefaultErrorProc retrieves, formats, and displays the error string corresponding to the given ICS error to the user. It is called for all errors that the application does not wish to handle itself.

Only severe, terminate, and warning errors are presented to the user unless the user runs the ICS Shell with the -d (for debug) parameter. In this case, all errors passed in to IcDefaultErrorProc are formatted and displayed to the user.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN The handle of the calling application's window.</td>
</tr>
<tr>
<td>hData</td>
<td>IN The handle of the open communication session for which the error occurred, or NULL if not applicable.</td>
</tr>
<tr>
<td>uType</td>
<td>IN The ICS error message type (for example, IC_ERROR, IC_RCVERROR, etc.) or NULL if not applicable.</td>
</tr>
<tr>
<td>error</td>
<td>IN The ICS error that occurred.</td>
</tr>
</tbody>
</table>
Return Value:

IC_OK is returned.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IC_RESULT data type
- IC_ERROR_INFO data type
- IC_ERROR_WARNING data type
- IC_ERROR_SEVERE data type
- IC_ERROR_TERMINATE data type
- IcGetString function
IcDeleteLibraryConfig

IC_RESULT FAR PASCAL IcDeleteLibraryConfig
    ( IC_RESULT_CONTEXT context,
    int TableName,
    int KeyIndex,
    void FAR * KeyStruct )

IcDeleteLibraryConfig deletes the record with the given key from the given table in
the ICS database.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>IN The library’s context.</td>
</tr>
<tr>
<td>TableName</td>
<td>IN The number of the table from which to delete.</td>
</tr>
<tr>
<td>KeyIndex</td>
<td>IN The zero-relative index of the key field from the beginning of the record.</td>
</tr>
<tr>
<td>KeyStruct</td>
<td>*IN A pointer to the key portion of the database table record structure for the given table with the necessary key field initialized.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. See Appendix C for other possible errors.
Notes:

- **IcDeleteLibraryConfig** is used only on library’s invisible tables (IC_TF_INVISIBLETABLE IC_TABLE_FLAGS flag). Path and Channel tables are managed by the ICS Manager and through the **IcLibUpdateConfig** procedure.

  ● WIN            ○ XVT            ○ DosLink

  ○ Accessory     ○ Shell          ○ Configurator

  ● AIL            ● SL             ● EIL

See also:

- **IcReadLibraryConfig** function
- **IcWriteLibraryConfig** function
- **IC_DICT_NODE** data type
- **IC_TABLE_FLAGS** data type
IcDeregisterAccessory

IC_RESULT FAR PASCAL IcDeregisterAccessory
( IC_RESULT_CONTEXT context )

IcDeregisterAccessory removes the association between the given IC_RESULT_CONTEXT and its accessory. The context is no longer valid.

Parameter     Description

context IN The IC_RESULT_CONTEXT of the accessory to deregister.

Return Value:
IC_OK is returned if successful, IC_ERROR_INTERNAL is returned if the context exceeds the context table bounds.

● WIN ○ XVT ○ DosLink
● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:
IC_RESULT_CONTEXT data type
IcRegisterAccessory function
IcDestroyHandle

void FAR PASCAL IcDestroyHandle(HANDLE hMem)

For ICS DosLink applications, IcDestroyHandle destroys the memory handle created by IcCreateHandle.

Parameter Description

hMem IN The memory handle to be destroyed.

Return Value:
None.

❍ WIN ❍ XVT ❍ DosLink

❍ Accessory ❍ Shell ❍ Configurator
❍ AIL ❍ SL ❍ EIL

See also:
IcCreateHandle function
**IcDestroyHwnd**

```c
void FAR PASCAL IcDestroyHwnd ( HWND hWnd )
```

For ICS DosLink applications, `IcDestroyHwnd` destroys the window handle created by `IcCreateHwnd`.

### Parameters

| hWnd | *IN | The ICS window handle to be destroyed. |

### Return Value:

None.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**

- `IcCreateHwnd` function
IcDestroySession

IC_RESULT FAR PASCAL IcDestroySession
   ( HIC_SESSION session )

For ICS DosLink applications, **IcDestroySession** destroys the ICS session structure created by **IcCreateSession**. The session handle is no longer valid, and all pending events for this session are destroyed. This implies that if the application uses **IcRegisterCallback**, it will no longer be called.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>IN</td>
</tr>
</tbody>
</table>

**Return Value:**

**IC_OK** is returned if successful. See Appendix C for possible errors.

**Note:** **IcDestroySession** must be called after **IcCloseSession** when the session handle is no longer needed. Failure to do so will result in a notification message from the DosLink.386 virtual device when the DOS virtual machine is destroyed stating that INFOConnect sessions were active.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**

- **IcCloseSession** function
- **IcOpenSession** function
IcDialogConfig

IC_RESULT FAR PASCAL IcDialogConfig
    ( HIC_CONFIG hConfig,
    HINSTANCE hInstance,
    LPCSTR Dlg,
    DLGPROC DlgProc,
    LPARAM lParam )

IcDialogConfig accesses the Windows DialogBoxParam procedure to display the given dialog box. Use it when you wish to display a dialog box for some given hConfig.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hConfig</td>
<td>IN The HIC_CONFIG handle of the open configuration session.</td>
</tr>
<tr>
<td>hInstance</td>
<td>IN The instance handle.</td>
</tr>
<tr>
<td>Dlg</td>
<td>*IN The dialog box template name.</td>
</tr>
<tr>
<td>DlgProc</td>
<td>IN The instance address of the dialog callback procedure.</td>
</tr>
<tr>
<td>lParam</td>
<td>IN The initialization value for lParam.</td>
</tr>
</tbody>
</table>

Return Value:
The return value is the MAKELONG of the value returned from the Windows DialogBoxParam function. For consistency, the dialog callback procedure can use EndDialog(..., LOWORD(IC_OK) for returning TRUE and EndDialog(..., LOWORD(IC_CANCELED) for returning FALSE.

See also:
IcLibUpdateConfig function
IcLibVerifyConfig function

WIN XVT DosLink
Accessory Shell Configurator
AIL SL EIL
IcExitOk

IC_RESULT FAR PASCAL IcExitOk ( BOOL Ok )

IcExitOk is used to notify INFOConnect Connectivity Services that a session can or cannot be closed. It is used in response to several IC_STATUS_COMMMGR status messages. A distributed application may use IcExitOk to prevent ICS from exiting in order to gracefully terminate the host component.

Parameters Description

Ok IN TRUE if the session may be safely closed, FALSE to abort the termination of ICS.

Return Value:

IC_OK if successful. See Appendix C for possible errors.

Note: If IcExitOk is not called in response to the IC_COMMMGR_QUERYEXIT status message, the ICS Shell will query the user for permission to close the open communication sessions.

See also:

IC_STATUS_COMMMGR data type
## IcFreeBuffer

IC_RESULT FAR PASCAL IcFreeBuffer
( HANDLE hBuffer )

IcFreeBuffer frees memory previously allocated through IcAllocBuffer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>The handle of the global buffer to free.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK if successful. See Appendix C for possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- AIL
- SL
- Configurator
- EIL

**See also:**

IcAllocBuffer function
IcGetBufferSize returns the size of the specified buffer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN The handle of a buffer allocated with IcAllocBuffer.</td>
</tr>
</tbody>
</table>

Return Value:

Size, in bytes, of the given memory block. If the given handle is not valid or if the memory has been discarded, this is zero.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IcAllocBuffer function
IcGetChannelID

IC_RESULT FAR PASCAL IcGetChannelID
  ( IC_RESULT_CONTEXT context,
    HIC_CHANNEL hIcChannel,
    LPSTR buffer,
    unsigned len )

IcGetChannelID obtains the channel ID from a channel handle.

Parameters | Description
---|---
context | IN A library context.
hIcChannel | IN The ICS HIC_CHANNEL handle of the channel from which to retrieve the channel ID.
buffer | *OUT A buffer to receive the channel ID.
len | IN The size of the buffer in bytes.

Return Value:
IC_OK if successful. IC_ERROR_BADPARAMETER if a parameter is incorrect.
IC_ERROR_TRUNCATED if the buffer was too small and the data was truncated.
See Appendix C for other possible errors.

WIN | XVT | DosLink
Accessory | Shell | Configurator
AIL | SL | EIL

See also:
HIC_CHANNEL data type
IcGetCmdlineOption

IC_RESULT FAR PASCAL IcGetCmdlineOption
   ( LPSTR sCmdLine,
     char option,
     char endDelimiter,
     LPSTR sValue,
     unsigned uValueSize )

IcGetCmdlineOption parses the given command line for the given option and retrieves the value associated with that option, if one exists. The option's value follows the option character on the command line.

IcGetCmdlineOption is always case INSENSITIVE.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sCmdLine</td>
<td>*IN The null terminated command line on which to parse.</td>
</tr>
<tr>
<td>option</td>
<td>IN The character option for which to search.</td>
</tr>
<tr>
<td>endDelimiter</td>
<td>IN The delimiter for the value of the option. This is usually a space.</td>
</tr>
<tr>
<td>sValue</td>
<td>*OUT The value associated with the given option, if one exists on the command line.</td>
</tr>
<tr>
<td>uValueSize</td>
<td>IN The size of the sValue buffer. It should be big enough to include an additional null character.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the option was found on the command line. In this case, sValue contains the value immediately following the option, if one exists. The IC_ERROR_NOFIND informational error is returned if the option was not found on the command line. IC_ERROR_TRUNCATED is returned if the destination buffer is too small for the option's value.

Notes:
INFOConnect API

- To access this procedure, include the icutil.h include file into your application.
- If the same option exists multiple times on the command line, IcGetCmdlineOption returns only the first occurrence.

● WIN ○ XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

Section 6, "ICS Accessory Definition"
**IcGetContext**

IC_RESULT FAR PASCAL IcGetContext
   ( LPSTR name,
     LPIC_RESULT_CONTEXT lpcontext )

IcGetContext provides the context associated with the given unique context string.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>*IN</td>
</tr>
<tr>
<td>lpcontext</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

A unique context identification string, as defined in the .HIC include file of the component.

An IC_RESULT_CONTEXT type that receives the context associated with `name`, if it exists.

**Return Value:**

IC_OK is returned if the context is found and returned.

IC_CONTEXTSTRING_NOT_FOUND is returned if the context could not be retrieved. In this case, the value pointed to by `lpcontext` is invalid.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**

LPIC_RESULT_CONTEXT data type

IcGetContextString function
IcGetContextID

IC_RESULT FAR PASCAL IcGetContextID
( LPSTR ID,
LPIC_RESULT_CONTEXT context )

IcGetContextID returns the context of the given library. The library is loaded, if necessary, and locked for use by the calling component. When the caller is done with the library, it must call IcReleaseContextID.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A library ID.</td>
</tr>
<tr>
<td>context</td>
<td>An IC_RESULT_CONTEXT to receive the library context.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. See Appendix C for possible errors.

WIN  ○ XVT  ○ DosLink

○ Accessory  ○ Shell  ● Configurator

● AIL  ● SL  ● EIL

See also:

IcReleaseContextID function
IcGetContextString

IC_RESULT FAR PASCAL IcGetContextString
  ( IC_RESULT_CONTEXT context,
    LPSTR buffer,
    unsigned length )

IcGetContextString provides the unique, null-terminated context string associated with the given context.

Parameters Description
context IN A context.
buffer *OUT A buffer to receive the unique context string associated with the given context.
length IN The size of the buffer in bytes.

Return Value:
IC_OK is returned if the context string is successfully retrieved. Otherwise, IC_CONTEXT_NOT_FOUND is returned and buffer is filled with NULLs.

WIN ● XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:
IC_RESULT_CONTEXT data type
IcGetContext function
ICGetINFOConnectDir

IC_RESULT FAR PASCAL ICGetINFOConnectDir
( enum IC_DIRECTORYTYPES dirtype,
LPSTR pstr,
unsigned strsize )

ICGetINFOConnectDir returns INFOConnect directory information.

Parameters Description

dirtype IN The IC_DIRECTORYTYPES type information to retrieve.
pstr *OUT The string to receive the information.
strsize IN The length of the string in bytes. This should be at least IC_MAXFILENAMESIZE.

Return Value:
IC_OK if successful. See Appendix C for possible errors.

Note: IC_CODEDIR requests the name of the directory containing the ICS code files. This directory can be a shared directory. IC_DATADIR requests the name of the directory containing the ICS data files. Applications should use this directory for all use configuration files.

● WIN ● XVT ○ DosLink

● Accessory ● Shell ● Configurator
○ AIL ○ SL ○ EIL

See also:
IC_DIRECTORYTYPES data type
IcGetLibraryDefault retrieves the default configuration data for the library's given table.

**Parameters**

- **context**: IN
  A library context.

- **TableNumber**: IN
  The number of the table for which to retrieve the default data.

- **buffer**: *OUT
  A buffer to receive the data.

- **len**: IN
  The size of the buffer in bytes.

**Return Value:**

- **IC_OK** if successful.
- **IC_ERROR_BADPARAMETER** if a parameter is incorrect.
- **IC_ERROR_TRUNCATED** if the buffer was too small and the data was truncated.

See Appendix C for other possible errors.
IcGetNewPath

IC_RESULT FAR PASCAL IcGetNewPath
   ( HANDLE hWnd,
     HANDLE hBuffer,
     unsigned len )

IcGetNewPath provides a programmatic interface to the ICS path configuration dialogs.

When the user has completed the configuration, an IC_NEWPATH message is sent to hWnd. At this point, the buffer designated by hBuffer will contain the unique, null-terminated path ID of the newly configured ICS path, or, if the user cancelled the path configuration, it will contain NULL.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN</td>
</tr>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
<tr>
<td>len</td>
<td>IN</td>
</tr>
</tbody>
</table>
Return Value:

IC_OK when the configuration procedure has been initiated.
IC_ERROR_BADPARAMETER (and the configuration procedures are not initiated) if len is less than IC_MAXPATHIDSIZE or if hBuffer is NULL.

● WIN ○ XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

IC_NEWPATH message
INFOConnect API

IcGetNextEvent

```c
void FAR PASCAL IcGetNextEvent
   ( HIC_SESSION session,
     LPHANDLE hWnd,
     LPWORD message,
     LPLONG lParam )
```

For ICS DosLink applications, IcGetNextEvent retrieves the next event for the session. It is used to poll for events, and may be used instead of, or in addition to, the callback routine.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>IN A session handle.</td>
</tr>
<tr>
<td>hWnd</td>
<td>*OUT The window handle on which the event occurred.</td>
</tr>
<tr>
<td>message</td>
<td>*OUT The event, or IC_NULLEVENT if no messages are available.</td>
</tr>
<tr>
<td>lParam</td>
<td>*OUT The long parameter for the event.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK is returned if successful. See Appendix C for possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

- AIL
- SL
- EIL
ICGetPathID

IC_RESULT FAR PASCAL IcGetPathID
( HIC_SESSION hsession,
  LPSTR buffer,
  unsigned length )

IcGetPathID provides the identification string of the ICS path for the given communication session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>An HIC_SESSION communication session handle. The session need not be established.</td>
</tr>
<tr>
<td>buffer</td>
<td>A global buffer to receive the null-terminated path identification string.</td>
</tr>
<tr>
<td>length</td>
<td>The size of the buffer in bytes. This must be at least IC_MAXPATHIDSIZE.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. Possible errors are IC_ERROR_BADPARAMETER and IC_ERROR_UNOPENEDSESSION. See Appendix C for other possible errors.
IcGetPathNames

IC_RESULT FAR PASCAL IcGetPathNames
    (HANDLE hBuffer,
      unsigned length)

IcGetPathNames provides a list of the configured path IDs. The list is returned in the given buffer and consists of a two-byte integer (count of configured ICS paths) followed by as many complete 'path entries' that will fit in the buffer. Each 'path entry' consists of a one byte (character) flag (‘1’ == currently active, ‘0’ == currently inactive) followed by a null-terminated ASCII string (the path ID).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>The handle to a buffer, allocated with IcAllocBuffer, in which the list is returned.</td>
</tr>
<tr>
<td>length</td>
<td>The size of the buffer in bytes.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. IC_ERROR_BADPARAMETER if len is less than 3 or if hBuffer is NULL. See Appendix C for other possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
IC_GET_RESULT_CONTEXT

IC_GET_RESULT_CONTEXT( result )

The IC_GET_RESULT_CONTEXT macro extracts the IC_RESULT_CONTEXT from the given IC_RESULT.

Parameters | Description
--- | ---
result | IN An IC_RESULT status or error from which the context is extracted.

Return Value:
The extracted context.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IC_RESULT data type
- IC_RESULT_CONTEXT data type
IC_GET_RESULT_SUBTYPE

IC_GET_RESULT_SUBTYPE ( result )

The IC_GET_RESULT_SUBTYPE macro extracts the IC_RESULT_SUBTYPE from the given IC_RESULT.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>IN</td>
</tr>
</tbody>
</table>

An IC_RESULT status or error from which the subtype is extracted.

Return Value:
The extracted subtype.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_RESULT data type
IC_RESULT_SUBTYPE data type
IC_GET_RESULT_SUBVALUE

IC_GET_RESULT_SUBVALUE ( result )

The IC_GET_RESULT_SUBVALUE macro extracts the IC_RESULT_SUBVALUE from the given IC_RESULT.

Parameters | Description
---|---
result | IN An IC_RESULT status or error from which the subvalue is extracted.

Return Value:
The extracted subvalue.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_RESULT data type
- IC_RESULT_SUBVALUE data type
IC_GET_RESULT_TYPE

IC_GET_RESULT_TYPE (result)

The IC_GET_RESULT_TYPE macro extracts the IC_RESULT_TYPE from the given IC_RESULT.

Parameters Description
result IN An IC_RESULT status or error from which the type is extracted.

Return Value:
The extracted type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_RESULT data type
- IC_RESULT_TYPE data type
IC_GET_RESULT_VALUE

IC_GET_RESULT_VALUE ( result )

The IC_GET_RESULT_VALUE macro extracts the IC_RESULT_VALUE from the given IC_RESULT.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>IN An IC_RESULT status or error from which the value is extracted.</td>
</tr>
</tbody>
</table>

Return Value:
The extracted value.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- El/L

See also:

- IC_RESULT data type
- IC_RESULT_VALUE data type
IcGetServiceName

IC_RESULT FAR PASCAL IcGetServiceName
    ( HIC_SESSION session,
    LPSTR name,
    unsigned length )

For ICS DosLink client/server applications, IcGetServiceName retrieves the service name of the partner session. If this is called by a server session, the pathname (that is, the path parameter from the client’s call to IcOpenSession) is returned.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session IN</td>
<td>A session handle.</td>
</tr>
<tr>
<td>name *OUT</td>
<td>A global buffer to receive the null-terminated service name.</td>
</tr>
<tr>
<td>length IN</td>
<td>The size of the buffer in bytes.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.

○ WIN ○ XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

IcOpenSession function
IcSetServerInfo function
IcGetSessionID

IC_RESULT FAR PASCAL IcGetSessionID
( HIC_SESSION hsession,
LPSTR buffer,
unsigned length )

IcGetSessionID returns the unique session identification string (ID) for the given session. The session ID consists of the path ID, followed by a semicolon and the unique session name, if it exists.

Parameters Description

hsession IN The HIC_SESSION handle of the communication session whose ID is to be retrieved.

buffer *OUT A buffer, allocated with IcAllocBuffer, in which to return the communication session ID.

length IN The size of the buffer in bytes. This must be at least IC_MAXSESSIONIDLEN.

Return Value:
IC_OK if successful. IC_ERROR_UNOPENEDSESSION if the session handle is invalid, IC_ERROR_TRUNCATED if the buffer was not large enough to hold the session ID. See Appendix C for other possible errors.
ICGetSessionInfo

IC_RESULT FAR PASCAL IcGetSessionInfo
   ( HIC_SESSION hsession,
   LPIC_SINFO info )

IcGetSessionInfo initializes the given IC_SINFO data structure with pertinent
information about the communication session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td>info</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if the structure was initialized. See Appendix C for possible errors.

- WIN
- XVT
- DosLink
- Accessory
  - Shell
  - Configurator
- AIL
  - SL
  - EIL

See also:

IC_SINFO data type
IcGetString

IcGetString FAR PASCAL IcGetString
  ( HIC_SESSION hsession,
   IC_RESULT result,
   LPSTR buffer,
   unsigned length )

IcGetString retrieves the text associated with the given error result. The null-terminated text is placed in the given buffer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>The communication session on which the error occurred, or NULL_HIC_SESSION if not relevant.</td>
</tr>
<tr>
<td>result</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>The error result.</td>
</tr>
<tr>
<td>buffer</td>
<td>*OUT</td>
</tr>
<tr>
<td></td>
<td>A buffer to receive the text.</td>
</tr>
<tr>
<td>length</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>The size of the buffer in bytes. This should be at least IC_MAXSTRINGLENGTH.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. See Appendix C for possible errors.

WIN XVT DosLink

Accessory Shell Configurator

AIL SL EIL

See also:

IC_RESULT data type
INFOConnect API

IcHandleOffset

IC_RESULT FAR PASCAL IcHandleOffset
    ( HIC_SESSION session,
    WORD utype,
    LPSTR mem,
    HANDLE FAR * lphandle )

For ICS DosLink applications, IcHandleOffset creates a memory handle for use with the ICS API.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>IN</td>
</tr>
<tr>
<td>utype</td>
<td>IN</td>
</tr>
<tr>
<td>mem</td>
<td>IN</td>
</tr>
<tr>
<td>lphandle</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.

Note: ICS DosLink applications would normally use IcAllocBuffer to obtain memory buffers. If the IcHandleOffset function is used, the call must immediately precede the call to IcRcv or IcXmt.

See also:

IcCreateHandle function
IcAllocBuffer function
IcInitIcs

IC_RESULT FAR PASCAL IcInitIcs
    ( int version,
      int revision )

IcInitIcs allows INFOConnect Connectivity Services to initialize, if necessary.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>IN</td>
</tr>
<tr>
<td>revision</td>
<td>IN</td>
</tr>
</tbody>
</table>

The highest ICS version that the calling program understands. The program does not take advantage of any new features that a higher ICS version may contain.

The highest ICS revision which the calling program understands. The program does not take advantage of any new features that a higher ICS revision may contain.

Return Value:

IC_OK if ICS initializes successfully or has been previously initialized,
IC_ERROR_NEWVERSION if a newer version of ICS is needed. See Appendix C for other possible errors.

Note: IcInitIcs MUST be called once prior to calling any of the INFOConnect Connectivity Services functions.

● WIN  ● XVT  ● DosLink

● Accessory
○ Shell
○ Configurator
○ AIL
○ SL
○ EIL

See also:

IC_STATUS_COMMMGR     data type
IC_STATUS              event
**IcIsDebug**

```
BOOL FAR PASCAL IcIsDebug
   ( enum IC_DEBUG debug )
```

*IcIsDebug* reports the status of the requested debug mode of INFOConnect.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug</code></td>
<td>IN</td>
</tr>
</tbody>
</table>

### Return Value:

The return value is TRUE if INFOConnect is running in the specified debug mode. FALSE otherwise.

- ● WIN
- ● XVT
- ○ DosLink
- ● Accessory
- ● Shell
- ○ Configurator
- ● AIL
- ● SL
- ● EIL

See also:

**IC_DEBUG** data type
**IClcl**

IC\_RESULT FAR PASCAL IcLcl

\[
(\text{HIC\_SESSION } hsession, \\
\text{IC\_LCL\_FLAGS } which )
\]

IcLcl cancels the pending request (designated by *which*) for the given communication session. An IC\_LCLRESULT message will be received for the cancelled requests.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hsession</em></td>
<td>IN  The established communication session's HIC_SESSION handle.</td>
</tr>
<tr>
<td><em>which</em></td>
<td>IN  An IC_LCL_FLAGS value that designates which pending request to cancel.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC\_OK is returned if the communication session is valid. Otherwise, IC\_ERROR\_UNOPENEDSESSION is returned. See Appendix C for other possible errors.

- WIN
- Accessory
- AIL
- XVT
- Shell
- SL
- DosLink
- Configurator
- EIL

**See also:**

IC\_LCL\_FLAGS data type
IcLibCloseChannel

IC_RESULT FAR PASCAL IcLibCloseChannel
( HIC_CHANNEL hLibChannel )

IcLibCloseChannel is provided by the ICS library and is called to terminate a communication channel. This routine is called after all sessions that were opened with this channel handle have been closed. At this point, channel related data may be cleaned up.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hLibChannel</td>
<td>IN The library handle of the channel to close. This is the value returned from the IcLibOpenChannel call.</td>
</tr>
</tbody>
</table>

Return Values:

IC_OK if successful. Otherwise, a standard or a library-specific error.

Notes:

- IcLibCloseChannel must be exported at ordinal value 17.
- If the library has no channel configuration information (that is, no IC_TF_CHANNELTABLE), IcLibCloseChannel is called once when the last session using the library is closed. The library should perform any session-related cleanup.

WIN ❍ XVT ❍ DosLink

❍ Accessory ❍ Shell ❍ Configurator

❍ AIL ❍ SL ❍ EIL

See also:

HIC_CHANNEL data type
IC_TABLE_FLAGS data type
IcLibOpenChannel function
IcLibCloseSession

IC_RESULT FAR PASCAL IcLibCloseSession
   ( HIC_SESSION hLibSession,
     HIC_CHANNEL hLibChannel )

IcLibCloseSession is provided by the ICS library and is called to terminate a
communication session. At this point, session related data is to be cleaned up.

Parameters Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hLibSession</td>
<td>IN The library handle of the session to close.</td>
</tr>
<tr>
<td>hLibChannel</td>
<td>IN The library handle of the session's channel.</td>
</tr>
</tbody>
</table>

Return Values:

IC_OK if successful. Otherwise, a standard or a library-specific error.

Note: IcLibCloseSession must be exported at ordinal value 2.

WIN ☐ XVT ☐ DosLink

☐ Accessory ☐ Shell ☐ Configurator

AIL ☐ SL ☐ EIL

See also:

HIC_SESSION data type
HIC_CHANNEL data type
IcLibOpenSession function
**IcLibEvent**

```pascal
IC_RESULT FAR PASCAL IcLibEvent
( UINT uType,
  HIC_SESSION hLibSession,
  GLOBALHANDLE hBuffer,
  UINT uSize )
```

**IcLibEvent** is provided by the ICS library and allows the library to process events directed to it. After any initial processing, the corresponding events are issued up the library stack by calling **IcMgrSendEvent**.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uType</code></td>
<td>IN An Event</td>
</tr>
<tr>
<td><code>hLibSession</code></td>
<td>IN The library handle of a session.</td>
</tr>
<tr>
<td><code>hBuffer</code></td>
<td>IN A handle to a global buffer or the HIWORD of an <strong>IC_RESULT</strong>, depending on <code>uType</code>.</td>
</tr>
<tr>
<td><code>uSize</code></td>
<td>IN The buffer size in bytes or the LOWORD of an <strong>IC_RESULT</strong>, depending on <code>uType</code>.</td>
</tr>
</tbody>
</table>

**Return Value:**

- **IC_OK** if the message is valid and can be processed for the given communication session. Otherwise, a standard or a library-specific error.

**Notes:**

- **IcLibEvent** must be exported at ordinal value 3.
- The first message received by the library is the **IC_SESSIONESTABLISHED** message. To guarantee that the session has been properly established, libraries must wait for this message before sending messages to the session.
- The last message received by the library is the **IC_SESSIONCLOSED** message. The library must not send any messages to the session after the **IC_SESSIONCLOSED** is received.
The library's session handle, hLibSession, may be NULL_HIC_SESSION when IcLibEvent is called with the IC_COMMMGR_INITIALIZED and IC_COMMMGR_TERMINATED status messages. When reacting to IC_COMMMGR_TERMINATED, the library may need to decrement its use count. If the use count is not zero when INFOConnect closes, an entry will be made into the trace log file by the active Trace library.

The AIL should both send the message to the application AND call IcMgrSendEvent. This allows the ICS Manager to verify that the ICS messages are flowing up the library stack.

• WIN  ○ XVT  ○ DosLink

○ Accessory  ○ Shell  ○ Configurator

• AIL  • SL  • EIL

See also:

HIC_SESSION data type
IcMgrSendEvent function
IcOpenSession function

Section 4, "ICS Messages/Events"
ICLibGetSessionInfo

IC_RESULT FAR PASCAL IcLibGetSessionInfo
  ( HIC_SESSION hLibSession,
    LPIC_SINFO sinfo )

ICLibGetSessionInfo is provided by the ICS library and is called to alter the pertinent fields in the given IC_SINFO record. An external interface library receives the structure with the ICS Manager defaults (currently, this is a zero-filled structure). The library must initialize all of the fields that pertain to it. A service library should modify only those fields that pertain to it.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hLibSession</td>
<td>IN</td>
</tr>
<tr>
<td>sinfo</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. Otherwise, a standard or a library-specific error.

Note: IcLibGetSessionInfo must be exported at ordinal value 10.

WIN XVT DosLink
Accessory Shell Configurator
AIL SL EIL

See also:
LPIC_SINFO data type
IC_SINFO data type
IcLibGetString

IC_RESULT FAR PASCAL IcLibGetString
    ( HIC_SESSION hLibSession,
    IC_RESULT result,
    LPSTR buffer,
    UINT length )

IcLibGetString is provided by the ICS library and should retrieve the null-terminated string associated with the given error result. Every library-specific error must have an associated string for displaying the error to the user.

If the library has used the IcSetSessionError utility, then the string may contain up to three string inserts (%s only). The ICS Manager will substitute the inserts into the string on behalf of the library, after the library returns.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| hLibSession | IN          | The library handle of the communication session on which the error occurred. The library may use information associated with this handle to modify the string before returning it.
|             |             | hLibSession is NULL_HIC_SESSION if the error is not associated with any session. |
| result      | IN          | The error result. |
| buffer      | *OUT        | A buffer to receive the string. |
| length      | IN          | The size of the buffer in bytes. |
INFOConnect API

Return Value:

IC_OK if successful. An IC_RESULT error otherwise. See Appendix C for possible errors.

Note: IcLibGetString must be exported at ordinal value 4.

WIN  XVT  DosLink
Accessory  Shell  Configurator
AIL  SL  EIL

See also:

IcSetSessionError function
IcLibIdentifySession

HANDLE FAR PASCAL IcLibIdentifySession
    ( HIC_SESSION hLibSession )

IcLibIdentifySession is provided by the ICS library. If all libraries in a session return \texttt{IC\_VERIFY\_OK} from \texttt{IcLibOpenSession(...IC\_OPEN\_VERIFY,...)}, \texttt{IcLibIdentifySession} is called for each library in the communication session to retrieve a unique session identifier.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hLibSession</td>
<td>IN  The library handle of a session.</td>
</tr>
</tbody>
</table>

**Return Value:**

If a library supports multiple sessions on a single path (a multiplexing library), it should return a handle to a global buffer (allocated through \texttt{IcAllocBuffer}) that contains a unique alphanumeric identification string. This string should meaningfully identify the session to the user. It may be up to \texttt{IC\_MAXSESSIONIDSUFFIX} bytes long. Additional bytes are truncated.

A library that is not multiplexing may return \texttt{(HANDLE)\_NULL}.

**Notes:**

- \texttt{IcLibIdentifySession} must be exported at ordinal value 5.
- \texttt{IcLibIdentifySession} is called starting with the service library at the top of the library stack and ending with the external interface library, until a single library returns a non-\texttt{NULL} buffer handle. If all libraries return \texttt{NULL}, the ICS Manager generates the unique session identification string using \texttt{hIcSession}.

\begin{itemize}
  \item WIN
  \item XVT
  \item DosLink
  \item Accessory
  \item Shell
  \item Configurator
  \item AIL
  \item SL
  \item EIL
\end{itemize}

**See also:**

\texttt{IcAllocBuffer} function
IcLibInstall

IC_RESULT FAR PASCAL IcLibInstall
  ( IC_RESULT_CONTEXT context )

IcLibInstall is provided by the ICS library and is called once by the ICS Manager
when the library is loaded for either configuration and/or communication session
establishment. It is used to initialize the library.

If installation fails (that is, returns an IC_ERROR_TERMINATE error type), the
library is immediately terminated. Therefore, only STANDARD terminate errors
may be returned from IcLibInstall in the failure case. Non-standard errors cannot be
used in the failure case because, since the library has not installed properly, it is not
available to return text through the IcLibGetString function.

Parameters | Description
---|---
context | IN The unique context identification for the library.

Return Value:

IC_OK if installation completes successfully. If installation fails, return a
STANDARD IC_RESULT terminate error. In this case, the library is immediately
unloaded. If a library-specific IC_ERROR_SEVERE is returned, the library is
loaded and IcLibGetString may be called to retrieve the text associated with the
error. In this case, no sessions will be opened over this library, and IcLibTerminate
will eventually be called before the library is unloaded. See Appendix C for possible
errors.
Notes:
- `IcLibInstall` must be exported at ordinal value 6.
- `IC_ERROR_INFO` and `IC_ERROR_WARNING` type return values do not constitute installation failure.

                   ● WIN    ○ XVT    ○ DosLink
                   ○ Accessory   ○ Shell    ○ Configurator
                   ● AIL    ● SL    ● EIL

See also:

`IcLibTerminate` function
INFOConnect API

IcLibLcl

IC_RESULT FAR PASCAL IcLibLcl
(HIC_SESSION hLibSession,
IC_LCL_FLAGS which )

IcLibLcl is provided by the ICS library and is called to stop reception of communication messages. Each library should do what is necessary to cancel pending requests. After processing, ALL libraries must pass the request to the underlying component by calling IcMgrLcl.

Parameters Description

hLibSession IN The library handle of a session.

which IN Bit flag designating which pending request to cancel. See IC_LCL_FLAGS data type.

Return Value:

IC_OK is returned if the communication session is valid and the command can be processed. Otherwise, a standard or a library-specific error.

Notes:

− IcLibLcl must be exported at ordinal value 7.

− The IC_LCL_CLOSESESSION type indicates an impending call to IcLibCloseSession. The library should not attempt to use any of this session’s buffers once IcLibLcl returns from being called with the IC_LCL_CLOSESESSION flag.

− The which flag contains bit fields. Therefore, use bit operators to test for the necessary request types. For example, (which & IC_LCL_RCV) is TRUE if the IC_LCL_RCV bit is set.
All libraries, including EILs, must call IcMgrLcl to inform the underlying components that the library has completed processing. If the EIL fails to call IcMgrLcl on the IC_LCL_CLOSESESSION flag, the IC_SESSIONCLOSED message will never be sent to the EIL's IcLibEvent procedure and the session will never close.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IcMgrLcl function
IC_LCL_FLAGS data type
ICLibOpenChannel

IC_RESULT FAR PASCAL IcLibOpenChannel
    ( HIC_CHANNEL hIcChannel,
      void FAR * buffer,
      UINT len,
      IC_OPEN_OPTIONS Options,
      LPHIC_CHANNEL lphLibChannel )

IcLibOpenChannel is provided by the ICS library and is called to initialize a library channel. It is used to create and initialize channel related data. IcLibOpenChannel is called once before any session that uses this channel is opened. It should open the requested channel.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hIcChannel</td>
<td>IN The ICS HIC_CHANNEL handle of the channel. The library must use this value on all calls from the library to the ICS Manager where a channel handle is required. See IcGetChannelID.</td>
</tr>
<tr>
<td>buffer</td>
<td>*IN The buffer of global data for that channel. The data in this buffer corresponds to the data defined in the library's IC_TF_CHANNELTABLE data dictionary.</td>
</tr>
<tr>
<td>len</td>
<td>IN The size of the buffer in bytes.</td>
</tr>
<tr>
<td>Options</td>
<td>IN IC_OPEN_VERIFY if the channel should only be verified for opening. That is, if the session can be opened, return IC_VERIFY_OK without actually performing the open request.</td>
</tr>
</tbody>
</table>
**INFOConnect API**

$lphLibChannel$ *IN*/*OUT  
If needed, the library should assign a value which uniquely identifies this channel within the library (the default value is $hlcChannel$). The value returned here is used on all future calls from the ICS Manager to the library to identify the channel. (For example, `IcLibOpenSession`, `IcLibCloseChannel`.) See the discussion in the *IDK Basic Developer's Guide* about 'aliasing'.

**Return Values:**

IC_OK, or IC_ERROR_INFO or IC_ERROR_WARNING result type, if the open was successful. IC_VERIFY_OK if the verify was successful. Otherwise, a standard or a library-specific error.
Notes:

- **IcLibOpenChannel** must be exported at ordinal value 16.
- If the library has no channel configuration information (that is, no `IC_TF_CHANNELTABLE`), **IcLibOpenChannel** is called once with `NULL_HIC_CHANNEL`, a NULL buffer, and zero length. The library should simply return a successful result.

<table>
<thead>
<tr>
<th>WIN</th>
<th>XVT</th>
<th>DosLink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessory</td>
<td>Shell</td>
<td>Configurator</td>
</tr>
<tr>
<td>AIL</td>
<td>SL</td>
<td>EIL</td>
</tr>
</tbody>
</table>

See also:

- **HIC_CHANNEL** data type
- **LPHIC_CHANNEL** data type
- **IcLibCloseChannel** function
- **IC_OPEN_OPTIONS** data type
- **IC_TABLE_FLAGS** data type
IcLibOpenSession is provided by the ICS library and is called either to initialize a library communication session or to verify that a session can be opened. It is to be used to create and initialize session related data.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hIcSession</td>
<td>The ICS HIC_SESSION handle of the session to open. The library must use this value on all calls from the library to the ICS Manager where a session handle is required. (For example, IcMgrXmt, IcMgrRcv, etc.)</td>
</tr>
<tr>
<td>hLibChannel</td>
<td>The library handle of the session's channel. This is the value returned from IcLibOpenChannel.</td>
</tr>
<tr>
<td>buffer</td>
<td>The buffer of path-specific data for that channel. The data in this buffer corresponds to the data defined in the library's IC_TF_PATHTABLE data dictionary.</td>
</tr>
<tr>
<td>len</td>
<td>The size of the buffer in bytes.</td>
</tr>
</tbody>
</table>
INFOConnect API

**Options**          IN  **IC_OPEN_VERIFY** flag if the session should only be verified for opening. That is, if the session can be opened, return **IC_VERIFY_OK** and do not open the session. Otherwise, return an error.

**lphLibSession**    *IN/*OUT  If needed, the library should assign a value which uniquely identifies this session within the library (the default value is **hIcSession**).

**Return Values:**

**IC_OK**, or **IC_ERROR_INFO** or **IC_ERROR_WARNING** result type, if the open was successful. **IC_VERIFY_OK** if the verify was successful. Otherwise, a standard or a library-specific error.

**Notes:**

− **IcLibOpenSession** must be exported at ordinal value 9.
− The **Options** flag contains bit fields. Therefore, use bit operators to test for the necessary request types. For example, (**Options & IC_OPEN_VERIFY**) is TRUE if the **IC_OPEN_VERIFY** bit is set.
− The value returned in **lphLibSession** is used as the **hLibSession** input value with all future calls from the ICS Manager to the library to identify the session. For example, **IcLibXmt, IcLibRcv, IcLibLcl, IcLibSetResult, IcLibGetSessionInfo, IcLibIdentifySession, IcLibCloseSession** will all be called with this session handle. **IcLibGetString** and **IcLibEvent** will be called with this session handle if the session handle applies. See **IcLibGetString** and **IcLibEvent** for more information.
− If the library has no path data (that is, no **IC_TF_PATHTABLE**), **IcLibOpenSession** is called with a NULL buffer and zero length.
After a successful return, the ICS Manager calls `IcLibOpenSession` again with `(Options = IC_OPEN_VERIFY)` and `(hIcSession = NULL_HIC_SESSION)` to determine if multiple instances of this path are supported.

The ICS Manager continues the path verification through the list of libraries in the stack. If at least one library in the stack returns something other than `IC_VERIFY_OK`, then this path is excluded from the Select Path dialog box. Otherwise (that is, all libraries in the stack return `IC_VERIFY_OK`) the `IcLibIdentifySession` function is called starting with the service library at the top of the library stack and ending with the external interface library, until a single library returns a non-NULL buffer handle. If all libraries return NULL, the ICS Manager generates the unique session identification string using `hIcSession`.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- HIC_SESSION
data type
- LPHIC_SESSION
data type
- HIC_CHANNEL
data type
- IcLibCloseSession
function
- IcLibIdentifySession
function
- IC_OPEN_OPTIONS
data type
- IC_TABLE_FLAGS
data type
IcLibPrintConfig

IC_RESULT FAR PASCAL IcLibPrintConfig
  ( UINT TableNumber,
    IC_PRINT_DETAIL detail,
    void FAR * buffer,
    UINT len,
    LPSTR print,
    UINT prlen )

IcLibPrintConfig is provided by the ICS library and is called to obtain a displayable string of library-specific configuration information.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableName</td>
<td>The number of the configuration table from the library's resource file.</td>
</tr>
<tr>
<td>detail</td>
<td>The amount of detail to include. Currently, only summary information is supported. Therefore, this is IC_PRINT_SUMMARY.</td>
</tr>
<tr>
<td>buffer</td>
<td>The buffer of data for summarizing.</td>
</tr>
<tr>
<td>len</td>
<td>The size of the buffer in bytes.</td>
</tr>
<tr>
<td>print</td>
<td>A string to receive the summarized data.</td>
</tr>
<tr>
<td>prlen</td>
<td>The size of the output string in bytes. When (detail == IC_PRINT_SUMMARY), prlen is at least IC_MAXPRINTSTRING large.</td>
</tr>
</tbody>
</table>
Return Values:

IC_OK if successful. Otherwise, a standard or a library-specific error.

Notes:

- **IcLibOpenSession** must be exported at ordinal value 15.
- The summarized data is brief, including only pertinent information. For example, the TTY EIL may return a buffer for display as follows.

  COM1,2400,7,1,E

- **IcLibOpenSession** must return an IC_ERROR_UNKNOWN_COMMAND result for all unknown detail values.

  ● WIN ☐ XVT ☐ DosLink
  ☐ Accessory ☐ Shell ☐ Configurator
  ● AIL ☐ SL ● EIL
ICLibRcv

IC_RESULT FAR PASCAL IcLibRcv
   ( HIC_SESSION hLibSession,
     HANDLE buffer,
     UINT length )

ICLibRcv is provided by the ICS library and is called to receive data into the specified buffer. Each library should do what is necessary to initiate a receive. A service library should eventually pass the request to the underlying library stack by calling IcMgrRcv.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hLibSession</td>
<td>IN The library handle of a session.</td>
</tr>
<tr>
<td>buffer</td>
<td>IN A handle to a global buffer in which data will be returned.</td>
</tr>
<tr>
<td>length</td>
<td>IN The size of buffer in bytes.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if the communication session is valid and the command can be processed. If the library supports a single receive request at a time and already has a request outstanding, it should return IC_ERROR_RCV_BUSY. Otherwise, a standard or a library-specific error.

Note: IcLibRcv must be exported at ordinal value 8.

See also:

ICMgrRcv function
IcLibSetResult

IcLibSetResult is provided by the ICS library and passes status and error information between the various INFOConnect communication layers. After any necessary processing, the IC_RESULT is passed back to the ICS Manager by calling IcMgrSetResult. The result of calling IcMgrSetResult is to be the return value from IcLibSetResult.

Parameters Description

hLibSession IN The library handle of a session.

uType IN The type of the IC_RESULT: IC_ERROR or IC_STATUS.

result IN The IC_RESULT message. See Appendix B for the defined ICS statuses. See Appendix C for the defined ICS errors.

Return Value:
If a processing error occurred, a standard or a library-specific error. Otherwise, the return value from the call to IcMgrSetResult.

Note: IcLibSetResult must be exported at ordinal value 11.

See also:
IC_STATUS message
IC_ERROR message
IcMgrSetResult function
IcLibTerminate

IC_RESULT FAR PASCAL IcLibTerminate
  ( void )

IcLibTerminate is provided by the ICS library and is called once by the ICS
Manager when all library sessions and channels have been closed and the library is
about to be unloaded. Any global cleanup may be done here.

Parameters | Description
------------|------------
None.       |            

Return Value:

IC_OK if successful. Otherwise, a standard or a library-specific error.

Note: IcLibTerminate must be exported at ordinal value 12.

WIN ❍ XVT ❍ DosLink

❍ Accessory ❍ Shell ❍ Configurator

❍ AIL       ❍ SL     ❍ EIL

See also:

IcLibInstall function
IcLibUpdateConfig

IcLibUpdateConfig is provided by the ICS library and should present a dialog box to the administrator, when appropriate (for example, during an Add or Modify action). Otherwise, the procedure may perform any desired data cleanup, etc.

Note that the TableNumber parameter determines what type of configuration is being performed: path specific or global. Therefore, the appropriate dialog box may be displayed.

IcDialogConfig is provided to display the dialog box for the hConfig.

Parameters Description

hConfig IN The HIC_CONFIG handle of the open configuration session.

TableNumber IN The number of the table from the library's resource file that is being configured.

buffer *IN/*OUT A buffer of data to be modified. Note that this may contain default data from the library's RC file.

len IN The size of the buffer in bytes.

Command IN An IC_COMMAND. This is the action that caused this function to be called.

Return Values:

IC_OK if successful. IC_CANCELED if the user canceled from the dialog. Otherwise, a standard or a library-specific error.

Notes:
INFOConnect API

- **IcLibUpdateConfig** must be exported at ordinal value 1. It should allow the user to update the given data and it should return an appropriate result. The ICS Manager will update the ICS database accordingly.

- **IcLibUpdateConfig** must return an **IC_ERROR_UNKNOWN_COMMAND** result for all unknown Commands.

- For **IC_TF_PATHTABLE** and **IC_TF_CHANNELTABLE** tables, the library receives only the data it defines in its data dictionary. It does not receive the path or channel keys that are added by ICS. For the exceptional case where the library wishes to access this information, it can do so as follows for the **IC_TF_PATHTABLE**:

  typedef struct {
      ...
  } MYLIBPATHCONFIG;

  typedef struct {
      PATHID pathID;
      CHANNELID channelID;
      MYLIBPATHCONFIG MyLibPathConfig;
  } PATHCONFIG

  typedef PATHCONFIG FAR*LPPATHCONFIG;

  Therefore, the PATHID would be referenced by:

  (LPPATHCONFIG)((LPSTR)buffer - sizeof(PATHID) - sizeof(CHANNELID));

  And the CHANNELID would be referenced by:

  (LPPATHCONFIG)((LPSTR)buffer - sizeof(CHANNELID));
The **IC_TF_CHANNELTABLE** access is similar:

```c
typedef struct {
   ...
} MYLIBCHANNELCONFIG;

typedef struct {
    CHANNELID channelID;
    MYLIBCHANNELCONFIG MyLibChannelConfig;
} CHANNELCONFIG;

typedef CHANNELCONFIG FAR* LPCHANNELCONFIG;
```

Therefore, the **CHANNELID** would be referenced by:

```
(LPCHANNELCONFIG)((LPSTR)buffer - sizeof(CHANNELID));
```

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- **IC_COMMAND** data type
- **IC_DICT_NODE** data type
- **IC_TABLE_FLAGS** data type
IcLibVerifyConfig

IC_RESULT FAR PASCAL IcLibVerifyConfig
   ( HIC_CONFIG hConfig,
     UINT TableNumber,
     LPSTR buffer,
     UINT len,
     IC_VERIFY Command )

IcLibVerifyConfig is provided by the ICS library to verify the contents of the
collection buffer. If Command is IC_VER_DISPLAY, errors are to be displayed
to the user. If Command is IC_VER_MODIFY, the erroneous data is to be presented
to the user for correction. Otherwise, an error is returned.

Parameters Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hConfig</td>
<td>IN</td>
<td>The HIC_CONFIG handle of the open configuration session.</td>
</tr>
<tr>
<td>TableNumber</td>
<td>IN</td>
<td>The number of the table from the resource file that is being verified.</td>
</tr>
<tr>
<td>buffer</td>
<td>*IN/*OUT</td>
<td>A buffer of configuration data to be verified, and possibly modified.</td>
</tr>
<tr>
<td>len</td>
<td>IN</td>
<td>The size of the buffer in bytes.</td>
</tr>
<tr>
<td>Command</td>
<td>IN</td>
<td>An IC_VERIFY command.</td>
</tr>
</tbody>
</table>

Return Values:

IC_OK if successful. Otherwise, a standard or a library-specific error.
Notes:

- **IcLibVerifyConfig** must be exported at ordinal value 14. It is used to allow the library to perform semantic checking on configuration data that has been input in an alternate manner. The library may wish to call this routine from the IcLibUpdateConfig routine to perform semantic checking on its configuration data.

- **IcDialogConfig** is provided to display the dialog box for the hConfig.

- **IcLibVerifyConfig** must return an **IC_ERROR_UNKNOWN_COMMAND** result for all unknown Commands.

- If the library changes a configuration record structure (thus incrementing the revision number of the data table), INFOConnect automatically performs the upgrade from the previous format to the new format. Data is copied field by field from a record in the old format to a new record in the updated format according to the field numbers. **IcLibVerifyConfig** is then called with the **IC_VER_UPGRADE** command so that the library can perform any necessary data conversions using **IC_UPGRADE_INFO**. Finally, **IcLibVerifyConfig** is called with the **IC_VER_SAVE** command so that the library can verify the data record before that record is save into the configuration database.

After all data records have been processed, quick configuration is invoked.

See the **IC_VERIFY** data type and the IDK Basic Developer’s Guide for more information.

See also:

- **IC_VERIFY** data type
- **IC_UPGRADE_INFO** data structure
ICLibXmt

IC_RESULT FAR PASCAL IcLibXmt
    ( HIC_SESSION hLibSession,
        HANDLE buffer,
        UINT length )

ICLibXmt is provided by the ICS library and is called to initiate transmission of a data buffer. A service library should eventually pass the request to the underlying library stack by calling IcMgrXmt.

Parameters Description
hLibSession IN The library handle of a session.
buffer IN A handle to a global buffer of data.
length IN The number of bytes to transmit.

Return Value:
IC_OK is returned if the communication session is valid and the command can be processed. If the library supports a single transmit request at a time and already has a request outstanding, it should return IC_ERROR_XMT_BUSY. Otherwise, a standard or a library-specific error.

Note: IcLibXmt must be exported at ordinal value 13.

WIN ○ XVT ○ DosLink

○ Accessory ○ Shell ○ Configurator

● AIL ○ SL ● EIL

See also:
IcMgrXmt function
IcLockBuffer

LPSTR FAR PASCAL IcLockBuffer
    ( HANDLE hBuffer )

IcLockBuffer locks memory previously created through IcAllocBuffer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
</tbody>
</table>

The handle of a global buffer, allocated with IcAllocBuffer, to lock.

Return Value:
An LPSTR type pointer to the locked block of memory or NULL if the memory handle is not valid.

WIN        V XVT
            DosLink

Accessory  Shell   Configurator
AIl        SL       EIL

See also:
IcAllocBuffer function
IcUnlockBuffer function
The `IC_MAKE_RESULT` macro creates an `IC_RESULT` from the given `IC_RESULT_CONTEXT`, `IC_RESULT_TYPE`, and `IC_RESULT_VALUE`.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>context</code></td>
<td>IN An <code>IC_RESULT_CONTEXT</code>.</td>
</tr>
<tr>
<td><code>type</code></td>
<td>IN An <code>IC_RESULT_TYPE</code>.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>IN An <code>IC_RESULT_VALUE</code>.</td>
</tr>
</tbody>
</table>

### Return Value:

The created `IC_RESULT` status or error.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
-AIL
- SL
- EIL

### See also:

- `IC_RESULT` data type
- `IC_RESULT_CONTEXT` data type
- `IC_RESULT_TYPE` data type
- `IC_RESULT_VALUE` data type
ICMgrEilEvent

IC_RESULT FAR PASCAL IcMgrEilEvent
    ( HIC_SESSION hIcSession,
      UINT uType,
      HANDLE hBuff,
      UINT uSize )

ICMgrEilEvent allows external interface libraries to post messages to their own
event procedure. This may be useful for processing interrupts as events.

Parameters  Description

hIcSession  IN  The ICS Manager's
               HIC_SESSION handle.

uType       IN  A message type.

hBuff       IN  A handle to a global buffer or the
               HIWORD of an IC_RESULT,
               depending on uType.

uSize       IN  The buffer size in bytes or the
               LOWORD of an IC_RESULT,
               depending on uType.

Return Value:

IC_OK is returned if the communication session is valid and the command can be
processed. See Appendix C for possible errors.

● WIN  ○ XVT  ○ DosLink

○ Accessory  ○ Shell  ○ Configurator
● AIL  ● SL  ● EIL

See also:

IclibEvent  function
ICMgrGetSessionInfo

IC_RESULT FAR PASCAL IcMgrGetSessionInfo
   ( HIC_SESSION hSession,
     LPIC_SINFO info )

ICMgrGetSessionInfo initializes the given IC_SINFO data structure with pertinent information about the lower communication session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hSession</td>
<td>IN</td>
</tr>
<tr>
<td>info</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

hSession
The ICS Manager’s HIC_SESSION handle of an established communication session.

info
An IC_SINFO record to be filled with communication session information.

Return Value:
IC_OK if the structure was initialized. See Appendix C for possible errors.

WIN
XVT
DosLink
Accessory
Shell
Configurator
AIL
SL
EIL

See also:
IC_SINFO data type
IcMgrLcl

IC_RESULT FAR PASCAL IcMgrLcl
  ( HIC_SESSION hIcSession,
    UINT which )

IcMgrLcl is an entry point into the underlying library stack to stop reception of
communication messages. The pending request or requests (designated by which) for
the given communication session are cancelled.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hIcSession</td>
<td>IN</td>
</tr>
<tr>
<td>which</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the communication session is valid and the command can be
processed. See Appendix C for possible errors.

Note: All libraries, including EILs, must call IcMgrLcl to inform the underlying
components that the library has completed processing. If the EIL fails to
call IcMgrLcl when the IC_LCL_CLOSESESSION flag is received in
IcLibLcl, the IC_SESSIONCLOSED message will never be sent to the EIL's
IcLibEvent procedure and the session will never close.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_LCLRESULT message
IC_LCL_FLAGS data type
IcMgrRcv

IC_RESULT FAR PASCAL IcMgrRcv
    ( HIC_SESSION hICSession,
      HANDLE buffer,
      UINT length )

IcMgrRcv is an entry point into the underlying library stack to request to receive data into the specified buffer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hICSession</td>
<td>IN The ICS Manager's HIC_SESSION handle.</td>
</tr>
<tr>
<td>buffer</td>
<td>IN A handle to a global buffer in which received data will be returned.</td>
</tr>
<tr>
<td>length</td>
<td>IN The size of the buffer in bytes.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the communication session is valid and the command can be processed. See Appendix C for possible errors.

- WIN  ○ XVT  ○ DosLink

- Accessory  ○ Shell  ○ Configurator

- AIL  ● SL  ○ EIL

See also:

IC_RCVDONE message
IC_RCVERROR message
# IcMgrSendEvent

IC_RESULT FAR PASCAL IcMgrSendEvent
    ( HIC_SESSION hIcSession,
    UINT uType,
    HANDLE hBuffer,
    UNIT uSize )

IcMgrSendEvent posts a message to the next higher layer in the library stack for the given communication session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hIcSession</td>
<td>IN      The ICS Manager's HIC_SESSION handle.</td>
</tr>
<tr>
<td>uType</td>
<td>IN      A message type.</td>
</tr>
<tr>
<td>hBuffer</td>
<td>IN      A handle to a global buffer or the HIWORD of an IC_RESULT.</td>
</tr>
<tr>
<td>uSize</td>
<td>IN      The size of the buffer in bytes or the LOWORD of an IC_RESULT.</td>
</tr>
</tbody>
</table>
INFOConnect API

Return Value:

IC_OK if the message is valid and was posted to the next higher layer in the library stack. See Appendix C for possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_ERROR message
IC_RCVDONE message
IC_RECVERROR message
IC_SESSIONESTABLISHED message
IC_STATUS message
IC_XMTDONE message
IC_XMTERROR message
IcMgrSetResult

IC_RESULT FAR PASCAL IcMgrSetResult
   ( HIC_SESSION hIcSession,
     UINT uType,
     IC_RESULT result )

IcMgrSetResult is an entry point into the underlying library stack to process status and error information.

Parameters               Description

   hIcSession      IN   An ICS Manager's HIC_SESSION handle.

   uType           IN   The type of the IC_RESULT:
                        IC_ERROR or IC_STATUS.

   result          IN   The IC_RESULT message. See Appendix B for the defined ICS statuses. See Appendix C for the defined ICS errors.

Return Value:

IC_OK if successful. An IC_RESULT error otherwise. See Appendix C for possible errors.

● WIN    ◑ XVT    ◑ DosLink

 ◑ Accessory    ◑ Shell    ◑ Configurator
● AIL    ◗ SL    ◗ EIL

See also:

IC_ERROR       message
IC_STATUS       message
IC_RESULT       data type
**IcMgrTraceBuffer**

IC_RESULT FAR PASCAL IcMgrTraceBuffer  
( IC_RESULT CONTEXT Context,  
   HIC_SESSION hIcSession,  
   UINT uType,  
   LPSTR Tag,  
   void FAR * Buffer,  
   UINT Len )

IcMgrTraceBuffer allows libraries to write a buffer of data to the trace.log debug file. The data is only written if tracing has been enabled for the given session.

The IcTrace hook library also uses IcMgrTraceBuffer to trace INFOConnect data communications by writing buffer contents to a trace file, trace.log, located in the DataDir directory.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>The library's context.</td>
</tr>
<tr>
<td>hIcSession</td>
<td>An ICS Manager's HIC_SESSION handle.</td>
</tr>
<tr>
<td>uType</td>
<td>A message type, or IC_NULLEVENT if not applicable.</td>
</tr>
<tr>
<td>Tag</td>
<td>An identifying string.</td>
</tr>
<tr>
<td>Buffer</td>
<td>A buffer of data to write.</td>
</tr>
<tr>
<td>Len</td>
<td>The size of the buffer, in bytes.</td>
</tr>
</tbody>
</table>
Return Value:

IC_OK is returned if the request is valid, whether or not the data is written to the file (since tracing may not be enabled). See Appendix C for possible errors.

● WIN  ○ XVT  ○ DosLink
● Accessory  ○ Shell  ○ Configurator
● AIL  ● SL  ● EIL

See also:

IC_DEBUG  data type
IcMgrTraceResult  function
IcMgrTraceResult

IC_RESULT FAR PASCAL IcMgrTraceResult
( IC_RESULT_CONTEXT Context,
  HIC_SESSION hIcSession,
  UINT uType,
  LPSTR Tag,
  IC_RESULT Result )

IcMgrTraceResult allows libraries to write an IC_RESULT to the trace.log debug file. The data is only written if tracing has been enabled for the given session.

The IcTrace hook library also uses IcMgrTraceResult to trace INFOConnect data communications by writing events and IC_RESULTs to a trace file, trace.log, located in the DataDir directory.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>IN</td>
</tr>
<tr>
<td>hIcSession</td>
<td>IN</td>
</tr>
<tr>
<td>uType</td>
<td>IN</td>
</tr>
<tr>
<td>Tag</td>
<td>*IN</td>
</tr>
<tr>
<td>Result</td>
<td>IN</td>
</tr>
</tbody>
</table>
Return Value:

IC_OK is returned if the request is valid, whether or not the data is written to the file (since tracing may not be enabled). See Appendix C for possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_DEBUG data type
IcMgrTraceResult function
ICMgrXmt

IC_RESULT FAR PASCAL IcMgrXmt  
  ( HIC_SESSION hIcSession,  
    HANDLE buffer,  
    UINT length )

ICMgrXmt is an entry point into the underlying library stack to initiate transmission of the given data buffer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hIcSession</td>
<td>IN The ICS Manager's HIC_SESSION handle.</td>
</tr>
<tr>
<td>buffer</td>
<td>IN A handle to a global buffer of data.</td>
</tr>
<tr>
<td>length</td>
<td>IN The number of bytes to transmit.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the communication session is valid and the command can be processed. See Appendix C for possible errors.

- ● WIN  ○ XVT  ○ DosLink
- ○ Accessory  ○ Shell  ○ Configurator
- ● AIL  ● SL  ○ EIL

See also:

IC_XMTDONE message
IC_XMTERROR message
IcNextEvent

```
IC_RESULT FAR PASCAL IcNextEvent
   ( HIC_SESSION session,
     IC_NEXTEVENT_FLAGS flags,
     WORD delay )
```

For ICS DosLink applications, IcNextEvent indicates that the callback routine is ready for the next event. It can also be used to set a timer and to query for events.

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>IN A session handle.</td>
</tr>
<tr>
<td>flags</td>
<td>IN IC_NEXTEVENT_FLAGS flags.</td>
</tr>
<tr>
<td>delay</td>
<td>IN If the IC_NEXTEVENT_TIMER flag is set, this specifies, in milliseconds, the amount of time elapsed before receiving a timer event (IC_TIMER).</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK is returned if successful, or if the IC_NEXTEVENT_CHECK flag is specified and there are no events in the queue. If the IC_NEXTEVENT_CHECK flag is specified and there are events in the queue, IC_INFO_QEVENT is returned. See Appendix C for possible errors.

**Notes:**

- IcNextEvent must be called by the callback routine with the (IC_NEXTEVENT_POP|IC_NEXTEVENT_READY) flags when it is done processing an event. This removes the event from the queue and informs ICS that the callback routine is ready to receive the next event. ICS DosLink applications that use the IcRegisterCallback function (in contrast to polling using IcGetNextEvent), must follow each call to all ICS APIs with a call to IcNextEvent with the IC_NEXTEVENT_READY flag.
- ICS DosLink applications may poll ICS for events instead of, or as well as, registering the callback routine. See IcGetNextEvent for more information.
- For the INFOConnect 2.0 release, the event returned after the given delay is IC_SETDONE, NOT IC_TIMER.
INFOConnect API

- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_NEXTEVENT_FLAGS  data types
IcNotifyConfig

IC_RESULT FAR PASCAL IcNotifyConfig
    ( IC_COMPONENT ComponentNum,
      UINT TableNum,
      UINT Message,
      IC_SERIALNUM SerialNum )

IcNotifyConfig notifies configuration windows of changes that have been made to
the configuration database.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComponentNum</td>
<td>The supplier-specific IC_COMPONENT of the component that owns the table that has been altered.</td>
</tr>
<tr>
<td>TableNum</td>
<td>The table number of the table that has been altered.</td>
</tr>
<tr>
<td>Message</td>
<td>The message indicating the change.</td>
</tr>
<tr>
<td>SerialNum</td>
<td>The one-relative index of the record that has been altered. The serial number key is IC_KEY_SERIALNUM.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.
INFOConnect API

**Note:** Libraries that support dynamic tables use `IcNotifyConfig` to ensure that the `IC_MSG_CONFIG` message types are distributed properly. All messages must be distributed for `IC_TF_DYNAMICTABLE` tables. For `IC_TF_ACTIVE...` tables, `IcNotifyConfig` should be called for the update message only. It should be called on a timer tick or after some maximum transaction count.

- WIN
- Accessories
- AIL

- XVT
- Shell
- SL

- DosLink
- Configurator
- EIL

See also:

- `IC_MSG_CONFIG` data type messages
- `IC_TABLE_FLAGS` data type
# IcOpenAccessory

IC_RESULT FAR PASCAL IcOpenAccessory
( HWND hWnd,
 LPSTR name
 LPSTR options,
 LPSTR sessionname,
 LPIC_SINFO sinfo,
 LPHIC_SESSION lphsession )

IcOpenAccessory allows an application to invoke an ICS accessory via dynamically created ICS paths linked with the LOCAL external interface library.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN The handle of the window attached to this communication session.</td>
</tr>
<tr>
<td>name</td>
<td>*IN The accessory ID. See Appendix A for ICS Standard IDs.</td>
</tr>
<tr>
<td>options</td>
<td>*IN A null-terminated string of command line options, excluding the path (-p) option. See Section 6 for information on command line options.</td>
</tr>
<tr>
<td>sessionname</td>
<td>*IN A null-terminated identification string (not necessarily unique) created by the application that names the newly created ICS paths. This is the name that is used to create the communication session name that is returned by a call to IcGetSessionName. This name appears in the title bar of the invoked accessory.</td>
</tr>
<tr>
<td>sinfo</td>
<td>*IN An IC_SINFO record that has been previously initialized, possibly by a call to IcGetSessionInfo.</td>
</tr>
</tbody>
</table>
INFOConnect API

`lpSession` *OUT* An **HIC_SESSION** to receive the communication session handle of the newly opened session.

**Return Value:**

**IC_OK** if successful. Possible error results are **IC_ERROR_NOMEMORY**, **IC_ERROR_BADPARAMETER**, **IC_ERROR_ACCESSORY_NOT_FOUND**, and **IC_ERROR_ACCESSORY_FAILED**. See Appendix C for other possible errors.
Notes:

- Since `IcOpenAccessory` calls `IcRunAccessory`, it supports the `-Wxy` window state command line option. This option determines the state of the accessory's window when it is executed by the ICS Manager. The valid values for `x` and `y` are as follows:

<table>
<thead>
<tr>
<th>x</th>
<th>meaning</th>
<th>y</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>normal</td>
<td>a</td>
<td>active</td>
</tr>
<tr>
<td>m</td>
<td>maximized</td>
<td>b</td>
<td>background</td>
</tr>
<tr>
<td>i</td>
<td>iconized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>hidden</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using any other values results in the return of an `IC_ERROR_INVALID_WINOPTION` error.

The default window state is normal and active. Invalid value combinations are hidden/active and maximized/background. These combinations result in the return of an `IC_ERROR_INVALID_WINCOMBO` error.

- To invoke an accessory without a communication session connection between the calling application and the accessory, use `IcRunAccessory`.

  - WIN
  - XVT
  - DosLink
  - Accessory
  - Shell
  - Configurator
  - AIL
  - SL
  - EIL

See also:

- `IcGetSessionName` function
- `IcGetSessionInfo` function
- `IcRunAccessory` function
- `IC_SINFO` data structure
ICOpenSession

IC_RESULT FAR PASCAL IcOpenSession
( HWND hWnd,
  LPSTR path,
  LPHIC_SESSION lphsession )

ICOpenSession requests the establishment of a logical communications connection, either within the system (that is, the ICS path uses the LOCAL external interface library) or to another computer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN</td>
</tr>
<tr>
<td>path</td>
<td>*IN</td>
</tr>
<tr>
<td>lphsession</td>
<td>*OUT</td>
</tr>
</tbody>
</table>
*IN/OUT  For ICS DosLink
Client/Server type
applications, this must be
initialized as the session
handle that is obtained from a
call to IcCreateSession.

Return Value:
If the request is valid, then either IC_OK or an IC_ERROR_WARNING or
IC_ERROR_INFO type error (other than IC_ERROR_CANCELOPEN) is returned
and the LPHIC_SESSION is set to a valid session handle and the communication
session becomes associated with the application's window. Otherwise, an error is
returned, the session handle is set to NULL_HIC_SESSION, and the connection is
not available. Some possible errors are IC_ERROR_NOMEMORY and the
informational error IC_ERROR_CANCELOPEN. See Appendix C for other
possible errors.
Notes:

- **IC_ERROR_CANCELOPEN** is an IC_ERROR_INFO error type that indicates that the user cancelled from the select path dialog box. For this special return value, the session handle is NULL_HIC_SESSION and no session is opened. Therefore, this return value should be treated as a special case return value from IcOpenSession.

- If either IC_OK or an IC_ERROR_WARNING or IC_ERROR_INFO type error (other than IC_ERROR_CANCELOPEN) is returned, the ICS message IC_SESSIONESTABLISHED will be sent to the application when the communication session establishes. The session handle is not valid unless the IC_SESSIONESTABLISHED event is received with an IC_OK result, or an IC_ERROR_INFO or IC_ERROR_WARNING result type. This handle should then be used with any other INFOConnect Connectivity Services function dealing with this communication session.

- If an IC_SESSIONESTABLISHED event is received with an IC_ERROR_SEVERE or IC_ERROR_TERMINATE error result, communication session establishment failed and the session handle is invalid. The communication session is to be closed immediately by calling IcCloseSession.

- If using IcRegisterMsgSession to register for messages and the hWnd and path parameters are both NULL, then the Windows desktop automatically becomes the parent window. To prevent this, call IcSelectPath to display the select path dialog box from your application.

- For ICS DosLink applications that are using the callback facility, the hWnd parameter that is input here is the same handle that is used as the window handle of the callback function.
INFOConnect API

● WIN ○ XVT ● DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

IC_SESSIONESTABLISHED message
IcCloseSession function
IcCreateSession function
IcRegisterMsgSession function
IcSelectPath function
INFOConnect API

IcRcv

IC_RESULT FAR PASCAL IcRcv
( HIC_SESSION hsession,
  HANDLE buffer,
  UINT length )

IcRcv is called to request a block of data for the given communication session. For most sessions, one receive request may be outstanding for a session at a time, with the subsequent receive request will result in an IC_ERROR_RCV_BUSY receive error.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td>buffer</td>
<td>IN</td>
</tr>
<tr>
<td>length</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:
IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned. See Appendix C for other possible errors.
Note: When the receive request is complete, an ICS message of either IC_RCVDONE or IC_RCVERROR (or "IC_RcvDone" or "IC_RcvError", as appropriate) will be sent to the application.

WIN                  XVT                  DosLink

Accessory        Shell        Configurator
AIL                SL            EIL

See also:

IC_RCVDONE          message
IC_RCVERROR         message
IcLcl                function
**ICReadBuffer**

IC_RESULT FAR PASCAL IcReadBuffer

( HANDLE hBuffer,
  UINT BufOffset,
  void FAR * Data,
  UINT DataOffset,
  UINT Len )

ICReadBuffer reads data from a buffer identified by a Windows HANDLE to a buffer identified by a far pointer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN The handle of the buffer from which to read the data.</td>
</tr>
<tr>
<td>BufOffset</td>
<td>IN The offset into the buffer designated by hBuffer of the data. This is usually zero.</td>
</tr>
<tr>
<td>Data</td>
<td>*OUT The buffer to receive the data.</td>
</tr>
<tr>
<td>DataOffset</td>
<td>IN The offset into the buffer where the data is read. This is usually zero.</td>
</tr>
<tr>
<td>Len</td>
<td>IN The number of bytes to read.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK if successful. IC_ERROR_NOMEMORY if the buffer could not be locked. See Appendix C for other possible errors.

- WIN
  - XVT
  - DosLink

- Accessory
  - Shell
  - Configurator

- AIL
  - SL
  - EIL

See also:

<table>
<thead>
<tr>
<th>IcWriteBuffer</th>
<th>function</th>
</tr>
</thead>
</table>
IcReadLibraryConfig

IC_RESULT FAR PASCAL IcReadLibraryConfig
( IC_RESULT_CONTEXT context,
  int TableNumber,
  int KeyIndex,
  void FAR * KeyStruct,
  void FAR * buffer,
  unsigned len )

IcReadLibraryConfig reads the record with the given key(s) from the given table. KeyStruct is a pointer to the key portion of the data dictionary record structure for the given table. The necessary key field must be initialized.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>IN   The library's context.</td>
</tr>
<tr>
<td>TableNumber</td>
<td>IN   The number of the table from which to read.</td>
</tr>
<tr>
<td>KeyIndex</td>
<td>IN   The zero-relative index of the key field from the beginning of the record.</td>
</tr>
<tr>
<td>KeyStruct</td>
<td>*IN  The key portion of the database table record structure for the given table with the necessary key field initialized.</td>
</tr>
<tr>
<td>buffer</td>
<td>*OUT A buffer to receive the record. This cannot contain the same structure pointed to by KeyStruct.</td>
</tr>
<tr>
<td>len</td>
<td>IN   The size of the buffer in bytes. This should be at least the size of the database record.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. IC_ERROR_TRUNCATED if the buffer was too small and the record was truncated. See Appendix C for other possible errors.

Notes:
INFOConnect API

- *IcReadLibraryConfig* is used only on library's invisible tables (*IC_TF_INVISIBLETABLE* flag). Path and Channel tables are managed by the ICS Manager and through *IcLibUpdateConfig* procedure.

- WIN  ○ XVT  ○ DosLink

- ○ Accessory  ○ Shell  ○ Configurator

● AIL  ● SL  ● EIL

See also:

- *IcWriteLibraryConfig* function
- *IcDeleteLibraryConfig* function
- *IC_DICT_NODE* data type
- *IC_TABLE_FLAGS* data type
IcReAllocBuffer

HANDLE FAR PASCAL IcReAllocBuffer
    ( HANDLE hBuffer,
      unsigned bufsize )

IcReAllocBuffer reallocates memory previously created through IcAllocBuffer.

Parameters | Description
---|---

hBuffer | IN  The handle of the global buffer to reallocate.
bufsize | IN   The new size, in bytes, of the reallocated buffer.

Return Value:
A buffer handle is returned if the memory was reallocated, (HANDLE)NULL otherwise.

WIN ☎️ XVT ✄ DosLink

Accessory ☎️ Shell ☎️ Configurator
AIL ☎️ SL ☎️ EIL

See also:
IcAllocBuffer function
IcRegisterAccessory

**Parameters**

- **name**: *IN A null-terminated, unique accessory context string.
- **types**: IN Reserved for future use. Must be zero.
- **context**: *OUT An **IC_RESULT_CONTEXT** that receives the context associated with **name**.

**Return Value:**

**IC_OK** is returned if successful. See Appendix C for possible errors.

- ● WIN
- ● XVT
- ○ DosLink
- ● Accessory
- ○ Shell
- ○ Configurator
- ○ AIL
- ○ SL
- ○ EIL

**See also:**

- **IC_RESULT_CONTEXT** data type
- **LPIC_RESULT_CONTEXT** data type
ICRegisterCallback

IC_RESULT FAR PASCAL IcRegisterCallback
   ( HIC_SESSION session,
   IC_CALLBACK cb )

For ICS DosLink applications, IcRegisterCallback registers, or updates, the
application's callback function with ICS. This callback routine will be called for
each event on that session.

Each ICS API call implies that the callback routine is not ready to receive events.
Therefore, the application must call IcNextEvent after each ICS API call in order to
notify ICS that the callback routine is ready.

Parameters  Description

session     IN     A session handle.

cb          *IN    The callback function.

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.

Notes:

− When the callback routine is done processing an event, it should call
  IcNextEvent with the (IC_NEXTEVENT_POP|IC_NEXTEVENT_READY) flags to remove the event from the queue and inform ICS that it is ready to
  receive the next event. ICS DosLink applications that use the
  IcRegisterCallback function (in contrast to polling using IcGetNextEvent), must
  follow each call to all ICS APIs with a call to IcNextEvent with the
  IC_NEXTEVENT_READY flag.

− ICS DosLink applications may poll ICS for events instead of, or as well as,
  registering the callback routine. See IcGetNextEvent for more information.
INFOConnect API

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

- IcNextEvent function
- IC_NEXTEVENT_FLAGS data type
- IC_CALLBACK data type
ICRegisterMsgSession

IC_RESULT FAR PASCAL IcRegisterMsgSession
    ( HIC_SESSION hIcSession,
    HWND hWnd
    UINT wParam,
    UINT MessageOffset,
    UINT MessageCount )

IcRegisterMsgSession registers the ICS messages with Windows on a per-session basis.

*MessageOffset* is specific to each application and can be different for each *hSession*. Developers may use WM_USER as the message offset. The message switch statement may then be coded using *MessageOffset+IC_NULLEVENT*, etc. See Section 4 for the sequence of messages.

The given window will only receive messages from *MessageOffset* to (*MessageOffset + MessageCount*). To stop receiving messages, call *IcRegisterMsgSession* with the *MessageCount* parameter *IC_NULLEVENT*.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hIcSession</em></td>
<td>IN The <strong>HIC_SESSION</strong> handle of the open session.</td>
</tr>
<tr>
<td><em>hWnd</em></td>
<td>IN The handle of the window to receive the ICS messages.</td>
</tr>
<tr>
<td><em>wParam</em></td>
<td>IN A value to be passed in as the <em>wParam</em> word parameter for every message. For example, this may be <em>hSession</em> or perhaps the ID of a control.</td>
</tr>
<tr>
<td><em>MessageOffset</em></td>
<td>IN The Windows message offset for the messages. This is usually, but not necessarily, WM_USER.</td>
</tr>
<tr>
<td><em>MessageCount</em></td>
<td>IN The number of messages to register. See Section 4 for the number of messages available.</td>
</tr>
</tbody>
</table>

Return Value:

*IC_OK* is returned if successful. See Appendix C for possible errors.
Notes:

- If MessageOffset is zero, messages are returned as in the ICS 2.0 release (that is, they are registered with Windows RegisterWindowMessage procedure).

- Call IcOpenSession with a NULL window handle. Note that if the path ID parameter to IcOpenSession is also NULL, then the Windows desktop automatically becomes the parent window. To prevent this, call IcSelectPath to display the select path dialog box from your application.

See also:

IcOpenSession function
IcSelectPath function
IcReleaseContextID

IC_RESULT FAR PASCAL IcReleaseContextID
( IC_RESULT_CONTEXT context )

IcReleaseContextID releases the context of the given library from configuration. The library is unlocked from configuration and unloaded, if necessary.

IcReleaseContextID must be called by the configuration accessory after either IcGetContextID or IcAddRefContextID has been called and the library configuration has been accessed. In other words, each time IcGetContextID is called, IcReleaseContextID must eventually be called.

Parameters | Description
---|---
context | IN An IC_RESULT_CONTEXT to be released.

Return Value:

IC_OK if successful. See Appendix C for other possible errors.

- WIN
- XVT
- Accessory
- Shell
- AIL
- SL
- Configurator
- DosLink
- EIL

See also:

IcGetContextID function
IcRunAccessory

IC_RESULT IcRunAccessory
( LPSTR ID,
  LPSTR options )

IcRunAccessory allows an application to invoke an ICS accessory. There is no communication session connection between the calling application and the accessory.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>*IN The accessory ID. See Appendix A for ICS Standard IDs.</td>
</tr>
<tr>
<td>options</td>
<td>*IN A null-terminated string of command line options. See Section 6 for information on command line options.</td>
</tr>
</tbody>
</table>

Return Value:
IC_OK if successful. Possible error results are IC_ERROR_NOMEMORY, IC_ERROR_BADPARAMETER, IC_ERROR_ACCESSORY_NOT_FOUND, and IC_ERROR_ACCESSORY_FAILED. See Appendix C for other possible errors.
Notes:

- **IcRunAccessory** supports the `-Wxy` window state command line option. This option determines the state of the accessory’s window when it is executed by the ICS Manager. The valid values for `x` and `y` are as follows:

<table>
<thead>
<tr>
<th>x</th>
<th>meaning</th>
<th>y</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>normal</td>
<td>a</td>
<td>active</td>
</tr>
<tr>
<td>m</td>
<td>maximized</td>
<td>b</td>
<td>background</td>
</tr>
<tr>
<td>i</td>
<td>iconized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>hidden</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using any other value results in the return of an **IC_ERROR_INVALID_WINOPTION** error.

The default window state is normal and active. Invalid value combinations are hidden/active and maximized/background. These combinations result in the return of an **IC_ERROR_INVALID_WINCOMBO** error.

- To invoke an accessory with a communication session connection between the calling application and the accessory (via dynamically created ICS paths linked with the LOCAL external interface library), use **IcOpenAccessory**.

  ● WIN      ● XVT  ○ DosLink
  ● Accessory ○ Shell ○ Configurator
  ○ AIL      ○ SL   ○ EIL

See also:

**IcOpenAccessory** function
ICRunHelp3 runs the Windows help system. The help file should be installed into the same directory as the executable, and it should have the same root file name as the library or accessory. For package component IDs, the package help file name is the last help file specified in the packages .INF installation file.

To access the INFOConnect help file, the type should be IC_MANAGER and ID and lpFile should be NULL.

If lpFile is a fully qualified file name, it is used as the help file. If the file name is not fully qualified or if it is NULL, the INFOConnect database will be queried to determine the fully qualified help file name. In this case, the type and ID must be given.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>IN</td>
</tr>
<tr>
<td>ID</td>
<td>*IN</td>
</tr>
<tr>
<td>lpFile</td>
<td>*IN</td>
</tr>
<tr>
<td>Topic</td>
<td>IN</td>
</tr>
</tbody>
</table>
Return Value:

IC_OK if successful. See Appendix C for possible errors.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL
**IcRunLibHelp**

```pascal
IC_RESULT FAR PASCAL IcRunLibHelp
   ( IC_RESULT_CONTEXT context,
    DWORD Topic )
```

**IcRunLibHelp** runs the Windows help system for the library with the given context ID. The help file should be installed into the same directory as the executable, and it should have the same root file name.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>context</code></td>
<td>IN</td>
</tr>
<tr>
<td><code>Topic</code></td>
<td>IN</td>
</tr>
</tbody>
</table>

**Return Value:**

**IC_OK** if successful. See Appendix C for possible errors.

- ● WIN
- ○ XVT
- ○ DosLink
- ○ Accessory
- ○ Shell
- ○ Configurator
- ● AIL
- ● SL
- ● EIL
**IcSelectPath**

IC_RESULT FAR PASCAL IcSelectPath
( HWND hWnd,
 HIC_CONFIG hPath,
 UINT Options,
 LPSTR PathID,
 UINT Len)

IcSelectPath displays the select path dialog box to the user, allowing the user to choose the path on which to open a session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN The window that becomes the parent window of the select path dialog box.</td>
</tr>
<tr>
<td>hPath</td>
<td>IN An HIC_CONFIG that is used to filter the available paths. Use NULL_HIC_CONFIG to present all available paths to the user.</td>
</tr>
<tr>
<td>Options</td>
<td>IN Use zero. This causes active paths to be excluded from the list of available paths.</td>
</tr>
<tr>
<td>PathID</td>
<td>*OUT The selected path ID, or NULL if none was selected.</td>
</tr>
<tr>
<td>Len</td>
<td>IN The size of the buffer. This should be at least IC_MAXPATHIDSIZE.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK if successful. The IC_ERROR_INFO error IC_ERROR_CANCELOPEN if the user cancelled from the dialog box. See Appendix C for a list of possible errors.
INFOConnect API

Note: Currently, the filtering of paths is not supported. Therefore, \textit{hPath} is always assumed to be \texttt{NULL_HIC_CONFIG}. 

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

\texttt{IcOpenSession} function
**ICSetError**

**IC_RESULT FAR PASCAL ICSetError**

( **HIC_SESSION hsession,**
  **IC_RESULT error**)  

**ICSetError** passes various error-type information through the INFOConnect communication session to a library in the library stack or to an attached application.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hsession</em></td>
<td>IN The communication session's handle.</td>
</tr>
<tr>
<td><em>error</em></td>
<td>IN The <strong>IC_RESULT</strong> error message. See Appendix C for the ICS errors.</td>
</tr>
</tbody>
</table>

**Return Value:**

**IC_OK** is returned if the communication session is valid. Otherwise, **IC_ERROR_UNOPENEDSESSION** is returned.

For ICS DosLink Client/Server applications, an **IC_ERROR_NOPARTNER** return value indicates that the other half of the session is not established and the request is ignored.

**Note:** Accessory-specific errors require a unique accessory context. To obtain this, use **IcRegisterAccessory**.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**

- **IC_RESULT** data type
- **IcRegisterAccessory** function
ICSetServerInfo

IC_RESULT  FAR PASCAL IcSetServerInfo
   ( HIC_SESSION session,
      LPIC_SINFO info )

For ICS DosLink applications, IcSetServerInfo makes the session a server session and initializes the given IC_SINFO data structure with pertinent information about the communication session. IcSetServerInfo must be called before calling IcOpenSession.

Once a session is declared as a server, the application may call IcSetServerInfo after calling IcOpenSession. This causes an IC_CONNECT_SERVER (IC_STATUS_CONNECT type) status to be sent to the client session.

Parameters Description

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>A session handle.</td>
</tr>
<tr>
<td>info</td>
<td>An IC_SINFO record to be passed to the client session during an IcGetSessionInfo call.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.

See also:

IcOpenSession function
IC_SINFO data type
IC_STATUS_CONNECT data type
IcSetSessionError

IC_RESULT FAR PASCAL IcSetSessionError
    ( HIC_SESSION hIcSession,
    IC_RESULT_CONTEXT context,
    IC_RESULT error,
    LPSTR lpinsert1,
    LPSTR lpinsert2,
    LPSTR lpinsert3 )

IcSetSessionError must be used when returning ICS standard error results (see Appendix C for a list of standard error results) to insure that the correct library is associated with the given error. The IcSetSessionError procedure may also be useful when returning library-specific errors, especially those which require up to three string inserts (%s formatting ONLY). This alleviates the library from managing the information itself.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hIcSession</td>
<td>IN The ICS session handle on which the error occurred, or NULL_HIC_SESSION if not applicable.</td>
</tr>
<tr>
<td>context</td>
<td>IN The unique library context.</td>
</tr>
<tr>
<td>error</td>
<td>IN The IC_RESULT error.</td>
</tr>
<tr>
<td>lpinsert1</td>
<td>*IN A string of maximum IC_MAXERRORINSERT bytes. This string will be used as the first string insert. The pointer itself may be NULL.</td>
</tr>
<tr>
<td>lpinsert2</td>
<td>*IN A string of maximum IC_MAXERRORINSERT bytes. This string will be used as the second string insert. The pointer itself may be NULL, and it must be if lpinsert1 is NULL.</td>
</tr>
</tbody>
</table>
INFOConnect API

lpinsert3 *IN

A string of maximum IC_MAXERRORINSERT bytes. This string will be used as the third string insert. The pointer itself may be NULL, and it must be if lpinsert2 is NULL.

Return Value:
The return value is the IC_RESULT error input parameter.

Example:
The following example shows returning the standard IC_ERROR_INTERNAL, which accepts one insert, a string describing the location of the error.

```
return IcSetSessionError(hIcSession, MyContext, IC_ERROR_INTERNAL, "IcLibOpenSession", NULL, NULL);
```

The following example returns a library-specific error, TTY_XMTERROR_TRANSMITTING.

```
IC_RESULT error;

error = IC_MAKE_RESULT(MyContext, TTY_ERROR, TTY_XMTERROR_TRANSMITTING);
return IcSetSessionError(hIcSession, MyContext, error, NULL, NULL, NULL);
```

See also:

IC_MAKE_RESULT function
IcLibGetString function
IcSetStatus

IC_RESULT  FAR  PASCAL  IcSetStatus
     (  HIC_SESSION  hsession,
          IC_RESULT  status )

IcSetStatus passes various status information through the INFOConnect communication session to a library in the library stack or to an attached application. An IC_STATUSRESULT message will be received when the status has been delivered.

Parameters Description

hsession  IN  The handle of the communication session.

status  IN  The IC_RESULT status message. See Appendix B for the defined ICS statuses.

Return Value:

IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENDSESSION is returned.

For ICS DosLink Client/Server applications, an IC_ERROR_NOPARTNER return value indicates that the other half of the session is not established and the request is ignored.

Note: Accessory-specific statuses require a unique accessory context. To obtain this, use IcRegisterAccessory.

WIN  XVT  DosLink

Accessory  Shell  Configurator

AIL  SL  EIL

See also:

IC_RESULT  data type
IcRegisterAccessory  function
ICUnlockBuffer

IC_RESULT FAR PASCAL ICUnlockBuffer
    ( HANDLE hBuffer )

ICUnlockBuffer unlocks memory previously locked by ICLockBuffer.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
</tbody>
</table>

The handle of a global buffer to unlock.

Return Value:

IC_OK if successful. See Appendix C for possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

ICAllocBuffer function
ICLockBuffer function
IcWriteBuffer

IC_RESULT FAR PASCAL IcWriteBuffer
  ( HANDLE hBuffer,
  UINT BufOffset,
  void FAR * Data,
  UINT DataOffset,
  UINT Len )

IcWriteBuffer writes data from a buffer identified by a far pointer to a buffer identified by a Windows HANDLE.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN/*OUT</td>
</tr>
<tr>
<td>BufOffset</td>
<td>IN</td>
</tr>
<tr>
<td>Data</td>
<td>*IN</td>
</tr>
<tr>
<td>DataOffset</td>
<td>IN</td>
</tr>
<tr>
<td>Len</td>
<td>IN</td>
</tr>
</tbody>
</table>
INFOConnect API

Return Value:

IC_OK if successful. IC_ERROR_NOMEMORY if the buffer could not be locked. See Appendix C for other possible errors.

● WIN ○ XVT ○ DosLink

● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL

See also:

IcReadBuffer function
**IcWriteLibraryConfig**

IC_RESULT FAR PASCAL IcWriteLibraryConfig
( IC_RESULT_CONTEXT context,
  int TableNumber,
  void FAR * buffer,
  unsigned len )

IcWriteLibraryConfig overwrites the given record in the given table in the ICS database. If the record does not exist, it is added; if the record does exist, it is located and updated using the primary key.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>context</code></td>
<td>IN The library's context.</td>
</tr>
<tr>
<td><code>TableNumber</code></td>
<td>IN The number of the table for which to write.</td>
</tr>
<tr>
<td><code>buffer</code></td>
<td>*IN The buffer to write.</td>
</tr>
<tr>
<td><code>len</code></td>
<td>IN The size of the buffer in bytes.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK if successful. IC_ERROR_INVALID_CONFIGREC if the length of the buffer does not equal the length of the record stored in the ICS database. See Appendix C for other possible errors.

**Notes:**

- IcWriteLibraryConfig is used only on library's invisible tables (IC_TF_INVISIBLETABLE flag). Path and Channel tables are managed by the ICS Manager and through IcLibUpdateConfig procedure.
INFOConnect API

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IcReadLibraryConfig function
IcDeleteLibraryConfig function
IC_DICT_NODE data type
IC_TABLE_FLAGS data type
IcXmt

IC_RESULT FAR PASCAL IcXmt
   ( HIC_SESSION hsession,
       HANDLE buffer,
       UINT length)

IcXmt is called to transmit a block of data for the given communication session. For most sessions, one transmission may be outstanding for a session at a time, with the subsequent transmit request will result in an IC_ERROR_XMT_BUSY transmit error.

If the IC_SINFO record indicates that the session is not transparent (that is, transparent == FALSE), then the data should NOT contain any special, protocol-specific characters. Special characters will be added by the underlying ICS libraries as required by the protocol.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td>buffer</td>
<td>IN</td>
</tr>
<tr>
<td>length</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned.
INFOConnect API

**Note:** When the transmission is complete, an INFOConnect Connectivity Services message of either **IC_XMTDONE** or **IC_XMTERRO** will be sent to the application. The buffer must not be modified until one of these messages is received or until an **IC_LCLRESULT** is received after a call to **IClcl**.

- **WIN**
  - **XVT**
  - **DosLink**

- **Accessory**
  - **Shell**
  - **Configurator**

- **AIL**
  - **SL**
  - **EIL**

**See also:**

- **IC_XMTDONE** message
- **IC_XMTERRO** message
- **IC_SINFO** data structure
The `NOREF` macro may be used to reference a procedure's formal parameter that would not otherwise be referenced. Unreferenced formal parameters cause nuisance errors from optimizing compilers.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>IN</td>
</tr>
</tbody>
</table>

**Return Value:**
The input variable.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
ic_buf_alloc

IC_BUFHND ic_buf_alloc
   ( long bufsize )

ic_buf_alloc allocates sharable memory and returns its IC_BUFHND handle type.

Parameter Description
bufsize IN The number of bytes to allocate.

Return Value:
An IC_BUFHND handle type is returned if the memory was allocated.
NULL_IC_BUFHND type is returned if the memory could not be allocated.

Note: ICS data communication buffers must be shared by different tasks. ic_buf_alloc ensures that these buffers are properly allocated to satisfy any operating system requirements for shared buffer. Therefore, buffers passed to the INFOConnect Connectivity Services routines MUST have been allocated through ic_buf_alloc.

See also:
ic_buf_free function
IC_BUFHND data type
NULL_IC_BUFHND data type
ic_buf_free

```c
void ic_buf_free
     ( IC_BUFHND hBuffer )
```

`ic_buf_free` frees memory previously allocated through `ic_buf_alloc`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hBuffer</code></td>
<td>IN The IC_BUFHND buffer memory handle of the buffer to free.</td>
</tr>
</tbody>
</table>

Return Value:

None.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- `ic_buf_alloc` function
- `IC_BUFHND` data type
ic_buf_lock

(1.0)

```c
STR_FAR ic_buf_lock
    (IC_BUFHND hBuffer)
```

ic_buf_lock locks memory previously created through ic_buf_alloc.

**Parameter** | **Description**
--- | ---
`hBuffer` | IN The IC_BUFHND buffer memory handle of the buffer to lock.

**Return Value:**
An XVT STR_FAR type pointer to the locked block of memory.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**
- ic_buf_alloc function
- ic_buf_unlock function
- IC_BUFHND data type
ic_buf_realloc

IC_BUFHND ic_buf_realloc
   ( IC_BUFHND hBuffer,
     long bufsize )

ic_buf_realloc reallocates memory previously created through ic_buf_alloc.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
<tr>
<td>bufsize</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

An IC_BUFHND handle type is returned if the memory was allocated.
NULL_IC_BUFHND type is returned if the memory could not be allocated.

See also:

ic_buf_alloc function
IC_BUFHND data type
NULL_IC_BUFHND data type
ic_buf_unlock

```c
void ic_buf_unlock
    ( IC_BUFHND hBuffer )
```

`ic_buf_unlock` unlocks memory previously locked through `ic_buf_lock`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
</tbody>
</table>

**Return Value:**
None.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**
- `ic_buf_lock` function
- `IC_BUFHND` data type
ic_change_handle

IC_RESULT ic_change_handle
    ( HIC_SESSION hsession,
      WINDOW hWnd )

ic_change_handle changes the ownership of a currently established communication session. All subsequent communication events are then directed to the main event procedure associated with that new window.

Parameters
hsession
    IN The HIC_SESSION handle of the opened communication session to which the new window becomes associated.

hWnd
    IN The WINDOW handle for the window that will obtain ownership of the given communication session.

Return Value:

IC_OK is returned if the change was successful.

IC_ERROR_UNOPENEDSESSION is returned if the given communication session is not a valid, established session. See Appendix C for other possible errors.

Note: An implicit ic_lcl( hs论述, IC_LCL_RCVXMT) is performed prior to the switch.

WIN  XVT  DosLink

Accessory  Shell  Configurator

AIL  SL  EIL

See also:
ic_lcl function
HIC_SESSION data type
IC_LCL_FLAGS data type
INFOConnect API

ic_close_session

IC_RESULT ic_close_session ( HIC_SESSION hsession )

ic_close_session causes INFOConnect Connectivity Services to close the given communication session.

Parameter Description
hsession IN The HIC_SESSION handle of the open communication session to close.

Return Value:
IC_OK is returned. The result of the communication session closure will be sent to the application's main event procedure through the INFOConnect-XVT event E_IC_SESSION_CLOSE. This result will be IC_OK if the communication session closed properly. See Appendix C for other possible errors.

Note: An IC_OK result from ic_open_session requires that ic_close_session be called regardless of the E_IC_SESSION_EST event result.

WIN ● XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

ic_open_session function
ic_default_error_proc

IC_RESULT ic_default_error_proc
    ( WINDOW hWnd,
      HIC_SESSION hsession,
      unsigned uType,
      IC_RESULT error )

ic_default_error_proc retrieves, formats, and displays the error string corresponding to the given ICS error to the user. It is called for all errors that the application does not wish to handle itself.

Only severe, terminate, and warning errors are presented to the user unless the user runs the ICS Shell with the -d (for debug) parameter. In this case, all errors that are passed in to this procedure are formatted and displayed to the user.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN The handle of the calling application's window.</td>
</tr>
<tr>
<td>hsession</td>
<td>IN The HIC_SESSION handle of the open communication session for which the error occurred, or NULL_HIC_SESSION if not applicable.</td>
</tr>
<tr>
<td>uType</td>
<td>IN The ICS error event type (for example, E_IC_ERROR, etc.) or NULL if not applicable.</td>
</tr>
<tr>
<td>error</td>
<td>IN The ICS error that occurred.</td>
</tr>
</tbody>
</table>
Return Value:
IC_OK is returned.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_RESULT data type
- HIC_SESSION data type
- NULL_HIC_SESSION data type
- IC_ERROR_INFO data type
- IC_ERROR_WARNING data type
- IC_ERROR_SEVERE data type
- IC_ERROR_TERMINATE data type
- ic_get_string function
ic_deregister_accessory

IC_RESULT ic_deregister_accessory (IC_RESULT_CONTEXT context)

ic_deregister_accessory removes the association between the given
IC_RESULT_CONTEXT and its accessory. The context is no longer valid.

Parameter Description
context IN The IC_RESULT_CONTEXT of
the accessory to deregister.

Return Value:
IC_OK is returned if successful, IC_ERROR_INTERNAL is returned if the context
exceeds the context table bounds.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
IC_RESULT_CONTEXT data type
ic_register_accessory function
ic_exit_ok

```
IC_RESULT ic_exit_ok
  ( BOOLEAN Ok )
```

`ic_exit_ok` is used to notify INFOConnect Connectivity Services that a session can or cannot be closed. It is used in response to several `IC_STATUS_COMMMGR` status messages. A distributed application may use `ic_exit_ok` to prevent ICS from exiting in order to gracefully terminate the host component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Ok</code></td>
<td>IN TRUE if the session may be safely closed, FALSE to abort the termination of ICS.</td>
</tr>
</tbody>
</table>

**Return Value:**

`IC_OK` if successful. See Appendix C for possible errors.

**Note:** If this procedure is not called in response to the `IC_COMMMGR_QUERYEXIT` status message, the ICS Shell will query the user for permission to close the open communication sessions.

- WIN
- ● XVT
- ○ DosLink
- ● Accessory
- ○ Shell
- ○ Configurator
- ○ AIL
- ○ SL
- ○ EIL

**See also:**

`IC_STATUS_COMMMGR` data type
ic_galloc

IC_MEMHND ic_galloc
   ( long bufsize )

ic_galloc allocates memory that is NOT sharable and returns its IC_MEMHND handle type.

Parameter Description
bufsize IN The number of bytes to allocate.

Return Value:
An IC_MEMHND handle type is returned if the memory was allocated.
NULL_IC_MEMHND type is returned if the memory could not be allocated.

Note: Buffers created through ic_galloc are for large, general purpose, intra-application memory usage. Use ic_buf_alloc for shared memory allocation.

WIN ● XVT ● DosLink
Accessory ● Shell ● Configurator
AIL ○ SL ○ EIL

See also:
ic_gfree function
IC_MEMHND data type
NULL_IC_MEMHND data type
**ic_get_context**

IC_RESULT ic_get_context  
( STR_FAR name,  
  LPIC_RESULT_CONTEXT lpcontext )

*ic_get_context* provides the context associated with the given unique context string.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>*IN The unique context string.</td>
</tr>
<tr>
<td>lpcontext</td>
<td>*OUT An IC_RESULT_CONTEXT type that receives the context associated with <em>name</em>, if it exists.</td>
</tr>
</tbody>
</table>

**Return Value:**

IC_OK is returned if the context is found and returned.  
IC_CONTEXTSTRING_NOT_FOUND is returned if the context could not be retrieved. In this case, the value pointed to by *lpcontext* is invalid.

- WIN
- ● XVT
- ○ DosLink
- ● Accessory
- ○ Shell
- ○ Configurator
- ○ AIL
- ○ SL
- ○ EIL

**See also:**

- LPIC_RESULT_CONTEXT data type
- ic_get_context_string function
ic_get_context_string

IC_RESULT ic_get_context_string
  ( IC_RESULT_CONTEXT context,
    STR_FAR buffer,
    unsigned length )

ic_get_context_string provides the unique, null-terminated context string associated with the given context.

Parameters | Description
---|---
context | IN A context.
buffer | *OUT A buffer to receive the unique context string associated with the given context.
length | IN The size of the buffer in bytes.

Return Value:
IC_OK is returned if the context string is successfully retrieved. Otherwise, IC_CONTEXT_NOT_FOUND is returned and buffer is filled with NULLs.

WIN ● XVT ● DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:
IC_RESULT_CONTEXT data type
ic_get_context function
ic_get_infoconnect_dir

IC_RESULT ic_get_infoconnect_dir
    ( enum IC_DIRECTORYTYPES dirtype,
      STR_FAR pstr,
      unsigned strsize )

ic_get_infoconnect_dir returns INFOConnect directory information.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirtype</td>
<td>IN The IC_DIRECTORYTYPES type of information to retrieve.</td>
</tr>
<tr>
<td>pstr</td>
<td>*OUT A string to receive the information.</td>
</tr>
<tr>
<td>strsize</td>
<td>IN The length of the string in bytes. This should be at least IC_MAXFILENAMESIZE.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. See Appendix C for possible errors.

Note: dirtype IC_CODEDIR requests the name of the directory containing the ICS code files. This directory can be a shared directory. IC_DATADIR requests the name of the directory containing the ICS data files. Applications should use this directory for all use configuration files.

○ WIN  ● XVT  ○ DosLink

● Accessory  ● Shell  ● Configurator
○ AIL  ○ SL  ○ EIL

See also:

IC_DIRECTORYTYPES        data type
ic_get_new_path

IC_RESULT ic_get_new_path
    ( WINDOW hWnd,
    IC_BUFHND hBuffer,
    unsigned len )

ic_get_new_path provides a programmatic interface to the ICS path configuration dialogs.

Parameters | Description
--- | ---
hWnd | IN The WINDOW handle of the calling application's window.
hBuffer | IN The handle to a globally allocated buffer to be filled with a null-terminated path identification (path ID) string. This buffer must have been allocated through ic_buf_alloc.
len | IN The size of buffer in bytes. This must be at least IC_MAXPATHIDSIZE.

Return Value:
IC_OK when the configuration procedure has been initiated.
IC_ERROR_BADPARAMETER (and the configuration procedure is not initiated)
if len is less than IC_MAXPATHIDSIZE or if hBuffer is NULL_IC_BUFHND.
INFOConnect API

Note: When the user has completed the configuration dialogs, an E_IC_NEWPATH event is sent to hWnd. At this point, the buffer designated by hBuffer will contain the unique, null-terminated path ID of the newly configured ICS path, or, if the user cancelled the path configuration, it will contain NULL.

- WIN  ● XVT  ○ DosLink

- Accessory  ○ Shell  ○ Configurator
- AIL  ○ SL  ○ EIL

See also:
E_IC_NEWPATH event
ic_get_path_id

IC_RESULT ic_get_path_id
    ( HIC_SESSION hsession,
    STR_FAR buffer,
    unsigned length )

ic_get_path_id provides the identification string of the ICS path for the given communication session.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN An HIC_SESSION communication session handle. The session need not be established.</td>
</tr>
<tr>
<td>buffer</td>
<td>*OUT A global buffer to receive the null-terminated path identification string.</td>
</tr>
<tr>
<td>length</td>
<td>IN The size of the buffer in bytes. This must be at least IC_MAXPATHIDSIZE.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. Possible errors are IC_ERROR_BADPARAMETER and IC_ERROR_UNOPENEDSESSION. See Appendix C for other possible errors.

○ WIN ● XVT ○ DosLink
○ Accessory ● Shell ○ Configurator
○ AIL ○ SL ○ EIL
ic_get_path_names

IC_RESULT ic_get_path_names
   ( IC_BUFHND buffer,
     unsigned length )

ic_get_path_names provides a list of the configured path IDs. The list is returned in
the given buffer and consists of a two-byte integer (count of configured path IDs)
followed by as many complete 'path entries' that will fit in the buffer. Each 'path
entry' consists of a one byte (character) flag ('1' == currently active, '0' == currently
inactive) followed by a null-terminated ASCII string (the path ID).

Parameters               Description

buffer              IN    A global buffer, allocated through
                      ic_buf_alloc, in which the list is
                      returned.

length             IN    The size of the buffer in bytes.

Return Value:
IC_OK if successful. IC_ERROR_BADPARAMETER if len is less than 3 or if
hBuffer is NULL_IC_BUFHND. See Appendix C for other possible errors.

● WIN     ● XVT     ● DosLink

● Accessory ● Shell    ● Configurator

● AIL      ● SL       ● EIL
ic_get_session_id

IC_RESULT ic_get_session_id
    ( HIC_SESSION hsession,
       STR_FAR buffer,
       unsigned length )

ic_get_session_id returns the unique session identification string (session ID) for the given session. The session ID consists of the path ID, followed by a semicolon and the unique session name, if it exists.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>The HIC_SESSION handle of the communication session whose ID is to be retrieved.</td>
</tr>
<tr>
<td>buffer</td>
<td>A global buffer, allocated with ic_buf_alloc, in which to return the communication session ID.</td>
</tr>
<tr>
<td>length</td>
<td>The size of the buffer in bytes. This must be at least IC_MAXSESSIONIDLEN.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. IC_ERROR_UNOPENEDSESSION if the session handle is invalid, IC_ERROR_TRUNCATED if the buffer was not large enough to hold the session ID. See Appendix C for other possible errors.
ic_get_session_info

IC_RESULT  ic_get_session_info
            ( HIC_SESSION  hsession,
              LPIC_SINFO  info )

ic_get_session_info initializes the given IC_SINFO data structure with pertinent information about the communication session.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td>info</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if the structure was initialized. See Appendix C for other possible errors.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_SINFO data type
ic_get_string

IC_RESULT ic_get_string
    ( HIC_SESSION hsession,
    IC_RESULT result,
    STR_FAR buffer,
    unsigned length )

ic_get_string retrieves the text associated with the given error result. The null-terminated text is placed in the given buffer.

Parameters Description
hsession IN The communication session on which the error occurred, or NULL_HIC_SESSION if not relevant.
result IN The error result.
buffer *OUT A buffer to receive the text.
length IN The size of the buffer in bytes. This should be at least IC_MAXSTRINGLENGTH.

Return Value:
IC_OK if successful. See Appendix C for possible errors.

See also:
IC_RESULT data type
ic_gfree

```c
void ic_gfree
    ( IC_MEMHND hBuffer )
```

**ic_gfree** frees memory previously allocated through **ic_galloc**.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
</tbody>
</table>

**IC_MEMHND** general memory handle of the buffer to free.

**Return Value:**

None.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**

- ic_galloc
- IC_MEMHND

**Function**

**Data type**
**ic_glock**

```c
STR_FAR ic_glock
    ( IC_MEMHND hBuffer )
```

`ic_glock` locks memory previously created through `ic_galloc`.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hBuffer</code></td>
<td>IN</td>
</tr>
</tbody>
</table>

The `IC_MEMHND` general memory handle of the buffer to lock.

**Return Value:**

An XVT STR_FAR type pointer to the locked block of memory.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**See also:**

- `ic_galloc` function
- `ic_gunlock` function
- `IC_MEMHND` data type
informationConnect API

ic_grealloc

IC_MEMHND ic_grealloc
    ( IC_MEMHND hnd,
      long bufsize )

ic_grealloc reallocates memory previously created through ic_galloc.

Parameters Description
hnd IN The IC_MEMHND general memory handle of the buffer to reallocate.
bufsize IN The new size, in bytes, of the reallocated buffer.

Return Value:
An IC_MEMHND handle type is returned if the memory was allocated.
NULL_IC_MEMHND type is returned if the memory could not be allocated.

See also:
- ic_galloc function
- IC_MEMHND data type
- NULL_IC_MEMHND data type
ic_gunlock

void ic_gunlock
    ( IC_MEMHND hBuffer )

ic_gunlock unlocks memory previously locked through ic_glock.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hBuffer</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

None.

See also:

ic_galloc function
ic_glock function
IC_MEMHND data type
NULL_IC_MEMHND data type
ic_init_ics

IC_RESULT ic_init_ics
   ( int version,
   int revision )

ic_init_ics allows INFOConnect Connectivity Services to initialize, if necessary. It MUST be called once from the application's appl_init routine prior to calling any of the INFOConnect Connectivity Services functions.

Parameters Description

version IN The highest ICS version which the calling program understands. The program does not take advantage of any new features that a higher ICS version may contain.

revision IN The highest ICS revision which the calling program understands. The program does not take advantage of any new features that a higher ICS revision may contain.

Return Value:
IC_OK if ICS initializes successfully or has been previously initialized,
IC_ERROR_NEWVERSION if a newer version of ICS is needed. See Appendix C for other possible errors.

○ WIN ● XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:
IC_STATUS_COMMMGR data type and
IC_STATUS event
ic_lcl

IC_RESULT ic_lcl
    ( HIC_SESSION hsession,
    short which )

ic_lcl cancels the pending request (designated by which) for the given communication session. An E_IC_LCL_RESULT event will be received for the cancelled requests.

Parameters | Description
--- | ---
hsession | IN The established communication session's HIC_SESSION handle type.
which | IN One of the IC_LCL_FLAGS values that designates which pending request to cancel.

Return Value:

IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned. See Appendix C for other possible errors.

oftware: WIN | XVT | DosLink

Accessory | Shell | Configurator

AIL | SL | EIL

See also:

IC_LCL_FLAGS data type
ic_open_accessory

IC_RESULT ic_open_accessory
    ( WINDOW hWnd,
      STR_FAR name,
      STR_FAR options,
      STR_FAR sessionname,
      LPIC_SINFO sinfo,
      LPHIC_SESSION lphsession )

ic_open_accessory allows an application to invoke an ICS accessory via
dynamically created ICS paths linked with the LOCAL external interface library.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN</td>
</tr>
<tr>
<td>name</td>
<td>*IN</td>
</tr>
<tr>
<td>options</td>
<td>*IN</td>
</tr>
<tr>
<td>sessionname</td>
<td>*IN</td>
</tr>
<tr>
<td>sinfo</td>
<td>*IN</td>
</tr>
</tbody>
</table>

hWnd: The XVT WINDOW handle of the window attached to this communication session.
name: The accessory ID. See Appendix A for ICS Standard IDs.
options: A null-terminated string of command line options, excluding the path (-p) option. (See Section 6 for information on command line options.)
sessionname: A null-terminated identification string (not necessarily unique) created by the application that names the newly created ICS paths. This is the name that is used to create the communication session name that is returned by a call to ic_get_session_id. This name appears in the title bar of the invoked accessory.
sinfo: An IC_SINFO record that has been previously initialized, possibly by a call to ic_get_session_info.
INFOConnect API

\[ lphsession \] *OUT An **HIC_SESSION** to receive the handle of the newly opened communication session.

**Return Value:**

**IC_OK** if successful. Possible error results are **IC_ERROR_NOMEMORY**, **IC_ERROR_BADPARAMETER**, **IC_ERROR_ACCESSORY_NOT_FOUND**, and **IC_ERROR_ACCESSORY_FAILED**. See Appendix C for other possible errors.

**Notes:**

− Since this procedure calls **IcRunAccessory**, it supports the \(-Wxy\) window state command line option. This option determines the state of the accessory’s window when it is executed by the ICS Manager. The valid values for \(x\) and \(y\) are as follows:

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td>n</td>
<td>normal</td>
</tr>
<tr>
<td>m</td>
<td>maximized</td>
</tr>
<tr>
<td>i</td>
<td>iconized</td>
</tr>
<tr>
<td>h</td>
<td>hidden</td>
</tr>
</tbody>
</table>

Using any other values results in the return of an **IC_ERROR_INVALID_WINOPTION** error.

The default window state is normal and active. Invalid value combinations are hidden/active and maximized/background. These combinations result in the return of an **IC_ERROR_INVALID_WINCOMBO** error.

− To invoke an accessory without a communication session connection between the calling application and the accessory, use **ic_run_accessory**.

○ **WIN** ○ **XVT** ○ **DosLink**

○ **Accessory** ○ **Shell** ○ **Configurator**

○ **AIL** ○ **SL** ○ **EIL**

**See also:**

**ic_get_session_id** function
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ic_get_session_info</td>
<td>function</td>
</tr>
<tr>
<td>IC_SINFO</td>
<td>data structure</td>
</tr>
</tbody>
</table>
ic_open_session

IC_RESULT ic_open_session
    ( WINDOW hWnd,
    STR_FAR path,
    LPHIC_SESSION lpsession )

ic_open_session requests the establishment of a logical communications connection either within the system (that is, the ICS path uses the LOCAL external interface library) or to another computer.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hWnd</td>
<td>IN</td>
</tr>
<tr>
<td>path</td>
<td>*IN</td>
</tr>
<tr>
<td>lpsession</td>
<td>*OUT</td>
</tr>
</tbody>
</table>

Return Value:

If the request is valid, then either IC_OK or an IC_ERROR_WARNING or IC_ERROR_INFO type error (other than IC_ERROR_CANCELOPEN) is returned, the LPHIC_SESSION is set to a valid HIC_SESSION handle and the communication session becomes associated with the application's window. Otherwise, an error is returned, the session handle is set to NULL_HIC_SESSION, and the connection is not available. Some possible errors are
IC_ERROR_NOMEMORY and the informational error IC_ERROR_CANCELOPEN. See Appendix C for other possible errors. See Appendix C for other possible errors.

Notes:

− **IC_ERROR_CANCELOPEN** is an IC_ERROR_INFO error type that indicates that the user cancelled from the select path dialog box. For this special return value, the session handle is **NULL_HIC_SESSION** and no session is opened. Therefore, this return value should be treated as a special case return value from ic_open_session.

− If either **IC_OK** or an IC_ERROR_WARNING or IC_ERROR_INFO type error (other than IC_ERROR_CANCELOPEN) is returned, the INFOConnect-XVT event **E_IC_SESSION_EST** will be sent to the application when the communication session establishes. The session handle is not valid unless the **E_IC_SESSION_EST** event is received with the event.v.ic.v.result of **IC_OK**, or with an IC_ERROR_INFO or IC_ERROR_WARNING result type. This handle should then be used with any other INFOConnect Connectivity Services function dealing with this communication session.

− If an **E_IC_SESSION_EST** event is received with an IC_ERROR_SEVERE or IC_ERROR_TERMINATE error result in event.v.ic.v.result, communication session establishment failed and the session handle type is invalid. The session is to be closed immediately by calling ic_close_session.

− If using ic_register_msg_session to register for messages and the hWnd and path parameters are both NULL, then the Windows desktop automatically becomes the parent window. To prevent this, call IcSelectPath to display the select path dialog box from your application.
INFOConnect API

- WIN
- • XVT
- • DosLink
- • Accessory
- • Shell
- • Configurator
- • AIL
- • SL
- • EIL

See also:

- E_IC_SESSION_EST
  - event
- ic_close_session
  - function
- ic_register_msg_session
  - function
- IcSelectPath
  - function
INFOConnect API

ic_rcv

IC_RESULT ic_rcv
  ( HIC_SESSION hsession,
    IC_BUFDND buffer,
    unsigned length )

ic_rcv is called to request a block of data for the given communication session. Only one receive request may be outstanding for a session at a time.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td>buffer</td>
<td>IN</td>
</tr>
<tr>
<td>length</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned. See Appendix C for other possible errors.

Note: When the receive request is complete, an INFOConnect-XVT event of either E_IC_RCV_DONE or E_IC_RCV_ERROR will be sent to the application.

○ WIN       ○ XVT       ○ DosLink
● Accessory ○ Shell     ○ Configurator
○ AIL       ○ SL        ○ EIL

See also:

E_IC_RCV_DONE  event
E_IC_RCV_ERROR event
ic_lcl         function
ic_register_accessory

IC_RESULT ic_register_accessory
( STR_FAR name,
unsigned types,
LPIC_RESULT_CONTEXT context )

ic_register_accessory associates the accessory name with a context, and returns that context through LPIC_RESULT_CONTEXT. The context is a dynamically assigned identification that can be used to uniquely identify the accessory when generating statuses and errors.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>*IN A null-terminated, unique accessory context string.</td>
</tr>
<tr>
<td>types</td>
<td>IN Reserved for future use. Must be zero.</td>
</tr>
<tr>
<td>context</td>
<td>*OUT An IC_RESULT_CONTEXT that receives the context associated with name.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.

WIN ● XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

IC_RESULT_CONTEXT data type
LPIC_RESULT_CONTEXT data type
IC_RESULT ic_register_msg_session
    ( HIC_SESSION hIcSession,
    WINDOW hWnd
    UINT wParam,
    UINT MessageCount )

ic_register_msg_session registers the ICS events with Windows on a per-session basis.

The given window will only receive events less than or equal to MessageCount. To stop receiving messages, call ic_register_msg_session with the MessageCount parameter E_IC_NULLEVENT.

Parameters Description

hIcSession IN The HIC_SESSION handle of the open session.

hWnd IN The handle of the window to receive the ICS messages.

wParam IN A value to be passed in as the event.v.ic.session word parameter for every message. For example, this may be hSession or perhaps the ID of a control.

MessageCount IN The number of messages to register. See Section 4 for the number of messages available.

Return Value:

IC_OK is returned if successful. See Appendix C for possible errors.
**Note:** Call `ic_open_session` with a NULL window handle. Note that if the path ID parameter to `ic_open_session` is also NULL, then the Windows desktop automatically becomes the parent window. To prevent this, call `IcSelectPath` to display the select path dialog box from your application.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

**See also:**
- `ic_open_session` function
- `IcSelectPath` function
ic_run_accessory

IC_RESULT  ic_run_accessory
           ( STR_FAR ID,
             STR_FAR options )

ic_run_accessory allows an application to invoke an ICS accessory. There is no communication session connection between the calling application and the accessory.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>*IN The accessory ID. See Appendix A for ICS Standard IDs.</td>
</tr>
<tr>
<td>options</td>
<td>*IN A null-terminated string of command line options. See Section 6 for valid command line options.</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK if successful. Possible error results are IC_ERROR_NOMEMORY, IC_ERROR_BADPARAMETER, IC_ERROR_ACCESSORY_NOT_FOUND, and IC_ERROR_ACCESSORY_FAILED. See Appendix C for other possible errors.
Notes:

- This procedure supports the -Wxy window state command line option. This option determines the state of the accessory’s window when it is executed by the ICS Manager. The valid values for x and y are as follows:

<table>
<thead>
<tr>
<th>x value</th>
<th>meaning</th>
<th>y value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>normal</td>
<td>a</td>
<td>active</td>
</tr>
<tr>
<td>m</td>
<td>maximized</td>
<td>b</td>
<td>background</td>
</tr>
<tr>
<td>i</td>
<td>iconized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>hidden</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using any other values results in the return of an IC_ERROR_INVALID_WINOPTION error.

The default window state is normal and active. Invalid value combinations are hidden/active and maximized/background. These combinations result in the return of an IC_ERROR_INVALID_WINCOMBO error.

- To invoke an accessory with a communication session connection between the calling application and the accessory (via dynamically created ICS paths linked with the LOCAL external interface library), use ic_open_accessory.

○ WIN  ● XVT  ○ DosLink

○ Accessory  ● Shell  ○ Configurator

○ AIL  ○ SL  ○ EIL

See also:

ic_open_accessory function
ic_set_error

IC_RESULT ic_set_error
    ( HIC_SESSION hsession,
    IC_RESULT error )

ic_set_error passes various error information through the INFOConnect communication session to a library in the library stack or to an attached application.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsession</td>
<td>IN</td>
</tr>
<tr>
<td>error</td>
<td>IN</td>
</tr>
</tbody>
</table>

Return Value:

IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned.

Note: Accessory-specific errors require a unique accessory context. To obtain this, use ic_register_accessory.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_RESULT data type

ic_register_accessory function
ic_set_status

IC_RESULT ic_set_status
    ( HIC_SESSION hsession,
    IC_RESULT status )

ic_set_status passes various status information through the INFOConnect communication session to a library in the library stack or to an attached application. An E_IC_STATUS_RESULT event will be received when the status has been delivered.

Parameters Description

hsession IN The session's handle.
status IN The IC_RESULT status message. See Appendix B for the defined ICS statuses.

Return Value:

IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned.

Note: Accessory-specific statuses require a unique accessory context. To obtain this, use ic_register_accessory.

See also:

IC_RESULT data type
ic_register_accessory function
ic_xmt

IC_RESULT ic_xmt
    ( HIC_SESSION hsession,
      IC_BUFHND buffer,
      unsigned length )

ic_xmt is called to transmit a block of data for the given communication session. Only one transmission may be outstanding for a session at a time. If the IC_SINFO record indicates that the session is not transparent (that is, transparent == FALSE), then the data should NOT contain any special, protocol-specific characters. Special characters will be added by the underlying ICS libraries as required by the protocol.

parameters          Description
hsession             IN    The established communication session's HIC_SESSION handle type.
buffer               IN    A buffer handle, allocated through ic_buf_alloc, of data to be transmitted.
length               IN    The number of bytes to transmit.

Return Value:
IC_OK is returned if the communication session is valid. Otherwise, IC_ERROR_UNOPENEDSESSION is returned.
Note: When the transmission is complete, an INFOConnect-XVT event of either E_IC_XMT_DONE or E_IC_XMT_ERROR will be sent to the application. The buffer must not be modified until one of these messages is received or until an E_IC_LCL_RESULT is received after a call to ic_lcl.

See also:

- E_IC_XMT_DONE event
- E_IC_XMT_ERROR event
- IC_SINFO data structure
Section 4
ICS Messages/Events

Several messages are generated in response to INFOConnect Connectivity Services events.

**MS-Windows Note**

To avoid potential conflicts with message numbers in MS-Windows, the messages must be registered with either `IcRegisterMsgSession` or `RegisterWindowMessage`. For `RegisterWindowMessage`, the message strings are designated in this section as quoted strings (like "IC_RcvDone"). See the *INFOConnect Basic Developer’s Guide* for information on registering Window’s messages.

Libraries use the defined message index. These are capitalized, such as `IC_RCVDONE`. 
Note

IcRegisterMsgSession and ic_register_msg_session ICS API are available to register messages using an offset and count. The following is the message sequence that may be used for the count of messages to register. Therefore, to register for all of the messages, use IC_LASTEVENT.

IC_NULLEVENT
IC_SESSIONESTABLISHED
IC_SESSIONCLOSED
IC_STATUS
IC_XMTDONE
IC_RCVDONE
IC_XMTERROR
IC_RCVERROR
IC_NEWPATH
IC_ERROR
IC_TIMER
IC_STATUSRESULT
IC_LCLRESULT
IC_SENDSTATUS
IC_LASTEVENT
E_IC_ERROR

This event is generated when an error occurs. The pertinent values in EVENT are as follows:

- `event.v.ic.session` is the session handle type for the communication session on which the error occurred or `NULL_HIC_SESSION` if not applicable.
- `event.v.ic.v.result` is the error result. See Appendix C for a list of ICS errors.

\[
\begin{align*}
\text{WIN} & \quad \text{XVT} & \quad \text{DosLink} \\
\text{Accessory} & \quad \text{Shell} & \quad \text{Configurator} \\
\text{AIL} & \quad \text{SL} & \quad \text{EIL}
\end{align*}
\]

See also:

EVENT data structure
ICS Messages/Events

**E_IC_LCL_RESULT**

This event is generated when a call to `ic_lcl` finally completes. The pertinent values in EVENT are as follows:

- **event.v.ic.session** is the session handle type for the communication session for which the message was generated.

- **event.v.ic.v.result** is the ICS result. See Appendix C for a list of ICS errors.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

- EVENT data structure
- ic_lcl function
E_IC_NEWPATH

This event is generated only when an application uses the `ic_get_new_path` interface into the ICS path configuration. It is sent to the application's main event function by the configuration accessory when the user exits the final configuration form. The pertinent values in EVENT are as follows:

- `event.v.ic.session` is the handle of the buffer that received the new path identification string.
- `event.v.ic.v.result` is `IC_OK` if the add succeeded, `IC_CANCELED` if the user cancelled from the dialogs, or from an ICS error. See Appendix C for a list of ICS errors.

**Note:**

This event is sent to the main configuration accessory's window to initiate the ICS path configuration. In this case, the pertinent values in EVENT are as follows:

- `event.v.ic.session` is the application's window handle to which the `E_IC_NEWPATH` event must be sent.
- `event.v.ic.v.rcv.buffer` is the handle of the buffer to receive the new path identification string.
- `event.v.ic.v.rcv.length` is the length, in bytes, of the buffer.

When processing has completed, the configuration accessory must post the `E_IC_NEWPATH` event to the application's window with the appropriate event values.

See also:

- EVENT data structure
- `ic_get_new_path` function
ICS Messages/Events

E_IC_NULLEVENT

This message indicates that no additional events are available.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
E_IC_RCV_DONE

This event is generated when a receive request completes. The pertinent values in EVENT are as follows:

- `event.v.ic.session` is the session handle type for the communication session for which the message was generated.
- `event.v.ic.v.rcv.buffer` is the buffer handle type of the received buffer.
- `event.v.ic.v.rcv.length` is the length, in bytes, of the buffer received.

See also:

- EVENT data structure
- ic_recv function
ICS Messages/Events

E_IC_RCV_ERROR

This event is generated when a receive request fails. The pertinent values in EVENT are as follows:

- `event.v.ic.session` is the communication session handle type.
- `event.v.ic.v.result` is the ICS error. See Appendix C for a list of ICS errors.

**Note:** Applications can ignore or log errors of type `IC_ERROR_INFO` or `IC_ERROR_WARNING` The receive request remains outstanding, which is in accordance with the definition of `IC_ERROR_INFO` and `IC_ERROR_WARNING` error types.

- WIN
  - XVT
  - DosLink

- Accessory
  - Shell
  - Configurator

- AIL
  - SL
  - EIL

See also:

- EVENT data structure
- ic_rcv function
E_IC_SESSION_CLOSE

This event is generated to notify an application of communication session termination. It normally results when the application requests session closure. However, it may also be caused by error conditions or by session termination by the user, such as clearing the session from the ICS Shell. The pertinent values in EVENT are as follows:

- `event.v.ic.session` is the handle type for the communication session for which the message was generated.

- `event.v.ic.v.result` is IC_OK if the communication session closed properly. Otherwise, it is an ICS error. See Appendix C for a list of ICS error results.

See also:

- EVENT data structure
- ic_close_session function
ICS Messages/Events

**E_IC_SESSION_EST**

This event is generated when a communication session is established as a result of a successful request to open a session. The pertinent values in EVENT are as follows:

- **event.v.ic.session** is the handle type for the communication session for which the message was generated.

- **event.v.ic.v.result** is **IC_OK** or an **IC_ERROR_INFO** or **IC_ERROR_WARNING** result type if the communication session establishment succeeded (this implies that the session handle type is now valid). Otherwise, the result is an ICS error. See Appendix C for a list of ICS errors.

**Note:** If the application receives the **E_IC_SESSION_EST** event with an **IC_ERROR_SEVERE** or **IC_ERROR_TERMINATE** result type, the session must be closed immediately. If the **E_IC_SESSION_EST** event result is an **IC_ERROR_INFO** or **IC_ERROR_WARNING** type (or **IC_OK**), the session may be used for communication before being closed.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

- EVENT data structure
- ic_open_session function
- ic_close_session function
E_IC_STATUS

This event is generated to report status information. The pertinent values in EVENT are as follows:

- **event.v.ic.session** is the session handle type for the communication session for which the message was generated.
- **event.v.ic.v.result** is the status. See Appendix B for the defined statuses.

**Notes:**

- The accessory need not process all status events.
- Applications that wish to react to **IC_COMMMGR_INITIALIZED** and **IC_COMMMGR_TERMINATED** statuses will receive **event.v.ic.session == NULL_HIC_SESSION** since these statuses are not associated with an INFOConnect session.
- This event reports status information that may have been generated by the underlying library stack, by the ICS Manager, or by another ICS application. In contrast, the **E_IC_STATUS_RESULT** event is received only after a call to **ic_set_status** and reports that the status event has been delivered.

○ WIN  ● XVT  ○ DosLink  ○ Accessory  ○ Shell  ○ Configurator  ○ AIL  ○ SL  ○ EIL

**See also:**

- EVENT data structure
E_IC_STATUS_RESULT

This event is generated when a call to ic_set_status finally completes. The pertinent values in EVENT are as follows:

- **event.v.ic.session** is the session handle type for the communication session for which the message was generated.
- **event.v.ic.v.result** is the ICS result. See Appendix C for a list of ICS errors.

=formatting not visible, appears as text: WIN XVT DosLink Accessory Shell Configurator AIL SL EIL

See also:

EVENT data structure

ic_set_status function
E_IC_XMT_DONE

This event is generated when a transmission request completes. The pertinent values in EVENT are as follows:

- `event.v.ic.session` is the session handle for the communication session for which the message was generated.
- `event.v.ic.v.rcv.buffer` is the handle of the transmitted buffer.
- `event.v.ic.v.rcv.length` is the length of the transmitted data.

See also:
- EVENT data structure
ICS Messages/Events

E_IC_XMT_ERROR

This event is generated if a transmission request fails. The pertinent values in EVENT are as follows:

− **event.v.ic.session** is the session handle type for the communication session for which the message was generated.

− **event.v.ic.v.result** is the ICS error. See Appendix C for a list of ICS errors.

**Note:** Applications may ignore or log errors of type **IC_ERROR_INFO** or **IC_ERROR_WARNING**. The transmit request remains outstanding, which is in accordance with the definition of **IC_ERROR_INFO** and **IC_ERROR_WARNING** error types.

○ WIN ● XVT ○ DosLink

● Accessory ○ Shell ○ Configurator
○ AIL ○ SL ○ EIL

See also:

EVENT data structure
IC_ERROR / "IC_Error"

This message is generated when an error, other than a transmit or receive error, occurs. The IC_RESULT accompanies this message.

The application receives the error in the lParam parameter. The HIC_SESSION on which the error occurred (or NULL_HIC_SESSION if not applicable) is returned to the application in wParam. Libraries may also generate library-specific errors which are indicated by the library's unique result context. See Appendix C for a list of standard errors.

WIN ○ XVT ● DosLink

○ Accessory ○ Shell ○ Configurator

● AIL ● SL ● EIL

See also:
IC_RESULT data type
IcMgrSendEvent function
IcMgrSetResult function
IC_LASTEVENT

This is the highest currently-defined INFOConnect message number. Use this to register for all ICS messages through **IcRegisterMsgSession**.

- **WIN**
- **XVT**
- **DosLink**
- **Accessory**
- **Shell**
- **Configurator**
- **AIL**
- **SL**
- **EIL**

**See also:**

**IcRegisterMsgSession** function
IC_LCLRESULT / "IC_LclResult"

This message is generated by an IIL when a call to IcLcl finally completes. The wParam data contains the handle for the communication session for which the message was generated. The lParam data contains the ICS result. See Appendix C for a list of ICS errors.

- WIN
- Accessory
- AIL

See also:

IcLcl function
ICS Messages/Events

IC_NEWPATH / "IC_NewPath"

This message is generated only when an application uses the IcGetNewPath interface into ICS path configuration. It is sent to the application by the configuration accessory when the user exits the last configuration form.

The application receives the handle of the buffer that received the new path identification string in wParam. The lParam data contains IC_OK if the add succeeded, IC_CANCELED if the user cancelled from the dialogs, or an ICS error. See Appendix C for a list of ICS errors.

Note:

This event is also sent to the main configuration accessory's window to initiate the ICS path configuration.

− The wParam data contains the application's window handle to which the IC_NewPath message must be sent.

− HI data of lParam contains the handle of the buffer designated to receive the new path identification string.

− LO data of lParam contains the length, in bytes, of the buffer.

When processing has completed, the configuration accessory must post the IC_NewPath message to the application's window with the appropriate values.

● WIN ○ XVT ○ DosLink

● Accessory ○ Shell  ● Configurator

○ AIL ○ SL ○ EIL

See also:

IcGetNewPath function
IC_NULLEVENT

This message indicates that no additional events are available. For ICS DosLink applications that are polling for messages, this is the message is generated when no other messages are available.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

- IcGetNextEvent function
ICS Messages/Events

IC_RCVDONE / "IC_RcvDone"

This message is generated when data becomes available due to the successful completion of a previous receive request. The buffer handle and the length of the received data accompany this message.

The application receives the handle for the communication session for which the message was generated in wParam. HI data of lParam contains the buffer handle of the received data. LO data of lParam contains the length of received data.

WIN

XVT

DosLink

Accessory

Shell

Configurator

AIL

SL

EIL

See also:

IcRcv function

IcMgrSendEvent function
IC_RCVERROR / "IC_RcvError"

This message is generated when a request to receive data fails. The IC_RESULT accompanies this message.

The application receives the handle for the communication session for which the message was generated in wParam. The lParam data contains the ICS error. See Appendix C for a list of ICS errors.

Notes:

− Applications can ignore or log errors of type IC_ERROR_INFO or IC_ERROR_WARNING. The receive request remains outstanding, which is in accordance with the definition of IC_ERROR_INFO and IC_ERROR_WARNING error types.

− Libraries should not generate IC_ERROR_INFO or IC_ERROR_WARNING type IC_RESULTs for this message in order to maintain compatibility with older ICS applications. Instead, these types of informational errors should be sent to the application through an IC_ERROR message. Note that the receive request remains outstanding, which is in accordance with the definition of IC_ERROR_INFO and IC_ERROR_WARNING error types.

If the library does generate informational and warning type receive errors, ICS 2.0 applications will need to execute with the Diagnostic service library to filter and translate these messages to IC_ERROR messages. For more information on the Diagnostic service library, see the IDK Developer's Guide.

− If a library also sends an IC_STATUS message to the application, the message should be sent after the error message so that the application is informed of the reason for the status message before the message is received.

● WIN ○ XVT ● DosLink

● Accessory ○ Shell ○ Configurator
● AIL ● SL ● EIL

See also:

IcRcv function
IcMgrSendEvent function
ICSENDSTATUS

This message is available for Interprocess Interface libraries to send status messages immediately up the library stack. Currently, the IC_STATUS_COMMMGR status messages are supported in this way. If the application has not registered for ICSENDSTATUS, the message is translated by the AIL to the IC_STATUS message before being delivered to the application.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_STATUS message
IC_SESSIONCLOSED / "IC_SessionClosed"

This message is generated to notify an application of communication session termination. It normally results from a request by the accessory to close the session. However, it may also be caused by error conditions or by user termination of a session.

The application receives the handle for the communication session for which the message was generated in the wParam data. The lParam data contains IC_OK if the communication session closed properly. Otherwise, it is an ICS error. See Appendix C for a list of ICS error results.

Notes:

− Only libraries with the max_version field of the IC_RC_NODE resource structure greater than IC_VERSION_2_0 will receive this message in their event procedures. The message must be passed up the library stack (by calling IcMgrSendEvent) in order for the session to close.

See also:

IC_CloseSession function
IC_SESSIONESTABLISHED / "IC_SessionEstablished"

This message is generated when a communication session finishes establishing as a result of a successful request to open a session.

The application receives the handle for the communication session for which the message was generated in the wParam data. The lParam data contains IC_OK or an IC_ERROR_INFO or IC_ERROR_WARNING result type if the communication session establishment succeeded (this implies that the session handle type is now valid). Otherwise, the lParam data contains an ICS error. See Appendix C for a list of ICS errors.

IC_SESSIONESTABLISHED is the first message received by a library, including EILs. To guarantee that the session has been properly established, libraries must wait for this message before sending any messages to the session or making any calls to IcMgrXmt, IcMgrRev, IcMgrLcl, or IcMgrSetResult. EIL developers must note that after any initial processing, this message must be issued up the library stack by calling IcMgrSendEvent.

Notes:

- If the application receives this message with an IC_ERROR_SEVERE or IC_ERROR_TERMINATE result type, the application must close the session immediately. If the message result is an IC_ERROR_INFO or IC_ERROR_WARNING type (or IC_OK), the session may be used for communication before being closed.

- Libraries with the max_version field of the IC_RC_NODE resource structure less than IC_VERSION_2_1 will receive ONLY IC_SESSIONESTABLISHED in their event procedures.
ICS Messages/Events

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IcOpenSession function
- IcCloseSession function
- IcLibEvent function
IC_MESSAGES / "IC_Status"

This message is generated to report status information from the underlying communication session layers. The IC_RESULT status accompanies this message.

The application receives the handle for the communication session for which the message was generated in the wParam data. The lParam data contains the specific status message. See Appendix B for defined status types.

Notes:

− The accessory need not process all status messages.

− Applications that wish to react to IC_COMMGR_INITIALIZED and IC_COMMGR_TERMINATED statuses must register "IC_Status" with Windows, since these statuses are not associated with an INFOConnect session (wParam == NULL_HIC_SESSION).

− Libraries with the max_version field of the IC_RC_NODE resource structure greater than IC_VERSION_2_0 will the IC_COMMGR_INITIALIZED and IC_COMMGR_TERMINATED statuses with hLibSession == NULL_HIC_SESSION, since these statuses are not associated with a session.

− This message reports status information that may have been generated by the underlying library stack, by the ICS Manager, or by another ICS application. In contrast, the IC_STATUSRESULT message is received only after a call to IcSetStatus and reports that the status message has been delivered.

● WIN ◦ XVT ● DosLink

● Accessory ◦ Shell ◦ Configurator
● AIL ● SL ● EIL

See also:

IcSetStatus function
IC_STATUSRESULT message
IcMgrSendEvent function
IcMgrSetResult function
IC_STATUSRESULT / "IC_StatusResult"

This message is generated by an IIL when a call to IcSetStatus finally completes. It indicates that the status has been delivered.

The application receives the handle for the communication session for which the message was generated in the wParam data. The lParam data contains the ICS result. See Appendix C for a list of ICS errors.

Note: The status messages that are delivered by calling IcSetStatus are delivered to the application via the IC_STATUS message.

See also: IcSetStatus function
IC_Status message
IC_MESSAGES/Events

IC_TIMER / "IC_Timer"

For ICS DosLink applications that use a callback routine and set a timer, this message is generated when the timer expires.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

IcNextEvent function
IC_XMTDONE / "IC_XmtDone"

This message is generated when a request to transmit data completes. The buffer handle of the transmitted data and the length of the transmitted data must accompany this message.

The application receives the handle for the communication session for which the message was generated in the wParam data. HI data of lParam contains the buffer handle of the transmitted data. LO data of lParam contains the length of the transmitted data.

- WIN
- Accessory
- AIL
- Oculus
- O V T
- SL
- Lo
- DosLink
- Shell
- Configurator
- EIL

See also:
- IcXmt function
- IcMgrSendEvent function
IC_XMTERROR / "IC_XmtError"

This message is generated if a request to transmit data fails. The IC_RESULT error accompanies this message.

The application receives the handle for the communication session for which the message was generated in the wParam data. The lParam data contains the error. See Appendix C for a list of ICS errors.

Notes:

− Applications can ignore or log errors of type IC_ERROR_INFO or IC_ERROR_WARNING. The transmit request remains outstanding, which is in accordance with the definition of IC_ERROR_INFO and IC_ERROR_WARNING error types.

− Libraries should not generate IC_ERROR_INFO or IC_ERROR_WARNING type IC_RESULTs for this message in order to maintain compatibility with older ICS applications. Instead, these types of informational errors should be sent to the application through an IC_ERROR message. Note that the transmit request remains outstanding, which is in accordance with the definition of IC_ERROR_INFO and IC_ERROR_WARNING error types.

If the library does generate informational and warning type transmit errors, ICS 2.0 applications will need to execute with the Diagnostic service library to filter and translate these messages to IC_ERROR messages. For more information on the Diagnostic service library, see the IDK Developer's Guide.

− If a library also sends an IC_STATUS message to the application, the message should be sent after the error message so that the application is informed of the reason for the status message before the message is received.

WIN ○ XVT ○ DosLink

■ Accessory ○ Shell ○ Configurator

■ AIL ■ SL ■ EIL

See also:

ICXmt function

ICMgrSendEvent function
The following data structures and types are defined for INFOConnect Connectivity Services.

**CHANNELID**

```c
typedef struct {
    char ID[IC_MAXCHANNELIDSIZE];
} CHANNELID;
```

This data structure type defines a channel identification string.

- **ID**
- The channel ID.

- ● WIN
- ● XVT
- ○ DosLink
- ○ Accessory
- ○ Shell
- ● Configurator
- ● AIL
- ● SL
- ● EIL
EVENT

typedef struct s_event {
  union {
    ...
    struct {
      HIC_SESSION session;
      union {
        struct { /* E_IC_RCV_DONE, E_IC_XMT_DONE */
          IC_BUFHND buffer;
          short length;
        } rcv;
        IC_RESULT result;
      } v;
    } ic;
    struct {
      char session;
      short function;
      IC_BUFHND datahnd;
      unsigned short length;
      unsigned short pspos
      BOOLEAN connected;
      IC_RESULT result;
    } sc;
    ...
  } v;
} EVENT, *EVENT_PTR;

This is an addition to the XVT EVENT structure. This addition occurs when the ICXVTMOD program is executed to update the XVT.H include file. See the INFOConnect Development Kit Developer's Guide for more information on using ICXVTMOD.

❖ WIN ❖ XVT ❖ DosLink
❖ Accessory ❖ Shell ❖ Configurator
❖ AIL ❖ SL ❖ EIL

See also:

Section 4, "ICS Messages/Events"
HIC_CHANNEL
A channel handle data type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

HIC_CONFIG
A configuration handle data type. A valid handle denotes a configuration session.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
IcOpen...Config functions

HIC_SESSION
A session handle data type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
ICS Data Structures/Types

**HIC_STATUSBUF**

An extended status buffer handle data type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IC_STATUSBUF data structure

**IC_BASEREVISION**

The base revision number of the **IC_BASEVERSION** of the ICS IDK that is supported by the ICS Manager.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**IC_BASEVERSION**

The base version number of the ICS IDK that is supported by the ICS Manager.
IC_BUFHND

INFOConnect Connectivity Services global buffer handle type for shared data.

WIN ● XVT ● DosLink

● Accessory ● Shell ● Configurator

● AIL ● SL ● EIL

IC_BUILD_REVISION

The specific generation number for this software revision. This number appears in parentheses at the end of the version string, IC_VERSION_STRING.

WIN ● XVT ● DosLink

● Accessory ● Shell ● Configurator

● AIL ● SL ● EIL

See also:

IC_VER_INFO data structure

IC_CALLBACK

typedef LONG

(FAR PASCAL *IC_CALLBACK)(WORD,WORD,WORD,LONG);

This is a special typedef for ICS DosLink applications that is used for the event callback routine.

WIN ● XVT ● DosLink

● Accessory ● Shell ● Configurator

● AIL ● SL ● EIL

See also:

IcRegisterCallback function
ICS Data Structures/Types

IC_COMMAND
ICS type used to communicate the action that is causing the library's IcLibUpdateConfig procedure to be invoked. The following commands are defined.

IC_CMD_ABOUT
The user is requesting About Box information.

IC_CMD_ADD
The user is performing an Add action.

IC_CMD_COPY
The user is performing a Copy action.

IC_CMD_DELETE
The user is performing a Delete action.

IC_CMD_DISCARD
This command is received when data from the previous call is being discarded.

IC_CMD_EXAMINE
The user is performing an Examine action.

IC_CMD_MODIFY
The user is performing a Modify action.

IC_CMD_SAVE
This command is received immediately before the data is saved to the database. If the data is not being saved, an IC_CMD_DISCARD command is received.

WIN  XVT  DosLink
Accessory  Shell  Configurator
AIL  SL  EIL

See also:
IcLibUpdateConfig function
IC_COMPONENT

ICS type that associates the supplier with the component. The IC_COMPONENT consists of a component number and a supplier number. Both generic and branded IC_COMPONENTs are defined. Generic IC_COMPONENTs are defined in ic.hic and are used by those components that conform to the interface defined in the component's .HIC include file. Branded IC_COMPONENTs encompass those components from a particular vendor. The supplier number of branded IC_COMPONENTs is assigned through the Malvern Development Group. The vendor is responsible for managing the component numbers for its INFOConnect products. The currently assigned supplier numbers, component numbers, and IC_COMPONENTs are recorded in the ic.hic include file for your reference. See Appendix A for more information.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- ic.hic header file
- Appendix A “Component Numbers

IC_COMPONENT_TYPE

Flags used in the component's resource file to mark the component's type.

IC_ACCESSORY

Accessory identification for use in the accessory resource file.

IC_APILIBRARY

Identification for use in the library resource file for the library component of an application. This library component should be installed either in the same directory as the ICS Manager (the default directory is the Windows system directory) or in the Windows DLL path. A library of this type will be accessed by various applications. It contains configuration information for the application and may perform configuration tasks and contain application-specific API. See IC_APPLIBRARY below.
ICS Data Structures/Types

**IC_APPINTERFACE**

Application Interface Library (AIL) identification for use in the library's resource file. Libraries of this type cannot be included in path templates. A library of this type creates sessions with itself as the topmost library. Also see **IC_IPCINTERFACE** and **IC_STACKINTERFACE** below.

**IC_APPLIBRARY**

Identification for use in the library resource file for the library component of an application. This library component should be installed in the same directory as the application and may contain application-specific API. A library of this type contains configuration information for the application and may perform configuration tasks. See **IC_APILIBRARY** above.

**IC_HOOKLIBRARY**

Hook Library identification for use in the resource file of a hook library. Hook libraries provide special features to the ICS Manager. The library must export the `IcLibInstall`, `IcLibTerminate`, and `IcLibGetString` procedures as documented.

**IC_INTERFACE**

External Interface Library (EIL) identification for use in the library's resource file.

**IC_IPCINTERFACE**

Interprocess Interface Library (IIL) identification for use in the library's resource file. IILs associate two sessions in different processes by internally linking the EIL role of one session to the AIL role of the other session. Libraries of this type are typically not included in path templates. This type of library is automatically included in sessions when an AIL requests a path that must be opened in a different process. The library acts as an EIL in the first session which it links to the second session where it acts in the AIL role.

**IC_LIBRARY**

Service or external interface library identification for use when the library type is not important.
ICS Manager identification.

**IC_QUICKCONFIG**

Quick Configuration Library identification for use in the resource file of a quick configuration library.

**IC_SERVICE**

Service Library identification for use in the library's resource file.

**IC_STACKINTERFACE**

Stacking interface library identification for use in the library's resource file. Libraries of this type typically implement multiplexing or switching functions on lower level sessions. Stack interface libraries associate two sessions in the same process by internally linking the EIL role of one session to the AIL role of the other session. Libraries of this type can be included in path templates as an EIL (for use by the higher level paths). During `IcLibOpenChannel` or `IcLibOpenSession`, this type of library typically behaves like an AIL and creates a session with a lower level path.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

**IC_RC_NODE** data structure
IC_DEBUG

ICS type used to set and query the mode of ICS. Tracing can occur at the application level (that is, the data flow between the application and the library stack is monitored), or it can occur at the library level (that is, the data flow between each library in the library stack is monitored, along with application level data flow). Enabling the tracing facility means that the Trace Service Library is inserted between as many libraries in the library stack as possible while maintaining a stack of less than the current 15 library limit.

IC_DEBUG_DBWINLOG

In this mode, the ICS Manager's debug window is enabled for displaying debug information if IC_DEBUG_TRACELOG is not set. If IC_DEBUG_TRACELOG is set, debug information is handled only by the TraceLog library.

IC_DEBUG_LIBOPENAPI

In this mode, the library open/close-type API are traced. This includes the install/terminate API, the open/close API for sessions, channels, etc., and the load/free library API.

IC_DEBUG_MODE

Implies that the default error procedure will display all messages that it receives to the user.

IC_DEBUG_MONITOR

In this mode, the library with the ID 'Monitor' is added to all sessions. The Monitor library maintains transaction-related information on a per-session basis.

IC_DEBUG_SWITCHES

This is the number of IC_DEBUG_... switches.

IC_DEBUG_TRACE

Enables tracing at the application level for all sessions that are opened with templates that have the Trace flag set.

IC_DEBUG_TRACEALL

Enables tracing at the application level for all sessions that are opened after this mode is set.
**IC_DEBUG_TRACEALLSTACK**

Enables tracing at the library level for all sessions that are opened after this mode is set.

**IC_DEBUG_TRACEENABLE**

Enables the addition of the library with the Trace library ID to be inserted in the stack of libraries.

**IC_DEBUG_TRACELOG**

Activates tracing. This indicates that the special purpose TraceLog library is loaded and tracing begins. The TraceLog library manages the trace log debug file for all library calls to `IcMgrTraceBuffer` and `IcMgrTraceResult`.

**IC_DEBUG_TRACEPATH**

Enables tracing at the application level for all sessions that are opened with paths that have the Trace flag set.

**IC_DEBUG_TRACEPATHSTACK**

Enables tracing at the library level for all sessions that are opened with paths that have the Trace flag set.

**IC_DEBUG_TRACESTACK**

Enables tracing at the library level for all sessions that are opened with templates that have the Trace flag set.

**Notes:**

- Note that application level tracing is controlled by:

  - **IC_DEBUG_TRACEALL**
    
    All open sessions
  
  - **IC_DEBUG_TRACEPATH**
    
    Open sessions with flagged paths (default set)
  
  - **IC_DEBUG_TRACE**
    
    Open sessions with flagged templates
Library level tracing, which includes application level tracing, is controlled by:

- **IC_DEBUG_TRACEALLSTACK**: All open sessions
- **IC_DEBUG_TRACEPATHSTACK**: Open sessions with flagged paths
- **IC_DEBUG_TRACESTACK**: Open sessions with flagged templates (default set)

See also:

- `IcIsDebug` function

### IC_DICT_FIELD

typedef struct aDictField {
  unsigned short StringId;
  unsigned short KeyFlags;
  unsigned short DataType;
  unsigned short BitOffset;
  unsigned short BitLength;
} IC_DICT_FIELD;

This data structure type defines the format of a single line in a data dictionary table. Each line of a data dictionary table describes a single field of a table record.

- **StringId**: The numeric ID of the string table entry for the string ID of this field. An ID of zero indicates the end of the data dictionary table.
- **KeyFlags**: The `IC_FIELD_FLAGS` flag that describes this field.
- **DataType**: The `IC_FIELDTYPE` that describes this field. The data type does not specify the length of the data. Length is specified by `BitLength`. 
**BitOffset**

The offset of this field, in bits, from the start of the structure. Using -1 causes the offset to automatically default to follow the previous field.

**BitLength**

The length of the field in bits.

**Note:** The length of a table record is computed by the field with the greatest \((\text{BitOffset} + \text{BitLength})\), converting this to byte size and rounding up to the nearest byte.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IC_FIELDTYPE
- IC_FIELD_FLAGS

### IC_DICT_NODE

typedef struct aICDICTNode {
    unsigned TableCount;
    unsigned TableFirst;
    unsigned DictRcType;
    unsigned DefaultRcType;
    unsigned reserved1;
    unsigned reserved2;
    unsigned reserved3;
    unsigned reserved4;
    IC_DICT_TABLE Table[];
} IC_DICT_NODE;

This data structure type defines the format of the `dictionary_id` RCDATA entry in the library's resource file. All libraries that contain path, channel, or invisible data must define that data in data dictionary tables. This resource entry is needed to access these tables. See **IC_DICT_FIELD** for the expected format of a line in a data dictionary table. Refer to Microsoft® Windows™ Software Development Kit Reference, User-Defined Resource Statement section for more information.

- **TableCount**
  
  The number of data dictionary tables.
TableFirst  The base number of the data dictionary tables.

DictRcType  The RC data type of the data dictionary tables.

DefaultRcType  The RC data type of the default data tables.
The numeric IDs of the default data tables must be the same as those of the corresponding data dictionary tables.

reserved1  Reserved, must be zero.

reserved2  Reserved, must be zero.

reserved3  Reserved, must be zero.

reserved4  Reserved, must be zero.

Table  See the IC_DICT_TABLE data structure.
Note that there must be TableCount Table entries.

Notes:

− The TableCount and TableFirst fields determine the data dictionary table numbers. Therefore, the tables must be numbered sequentially.

− Each data dictionary table must fully define the corresponding configuration data structure.

− Each data dictionary table must have a default data table that must have the same numeric ID as the corresponding data dictionary table. The default data table must contain the default data as a binary image of the corresponding data structure. No field-type processing is performed on this default, user-defined data resource. Therefore, it is imperative that the default data table be a binary image of the data dictionary table. Again, refer to Microsoft® Windows™ Software Development Kit Reference, User-Defined Resource Statement section and the INFOConnect Development Kit Basic Developer's Guide for more information.

● WIN  ● XVT  ○ DosLink

○ Accessory  ○ Shell  ● Configurator

● AIL  ● SL  ● EIL

See also:

IC_RC_NODE  data structure
IC_DICT_TABLE

typedef struct {
    IC_TABLE_FLAGS Flags;
    UINT TableId;
    UINT Ver;
} IC_DICT_TABLE;

This data structure type defines a table entry in the IC_DICT_NODE structure. The IC_DICT_NODE structure must contain one entry for each data dictionary table defined by the library.

Flags
The IC_TABLE_FLAGS for the table.

TableId
The string resource string number of the table's string identifier, or title.

Ver
The table version. Changes to Ver (in combination with the record length) between library releases identifies that the configuration database table requires reorganization.
ICS Data Structures/Types

See also:

IC_TABLE_FLAGS data type
IC_DICT_NODE data structure

IC_DIRECTORYTYPES
ICS type relating to directory-type and database-type information.

IC_CODEDIR
This is the directory type for retrieving the name of the directory that contains INFOConnect code files. It is either the CodeDir entry from the [INFOConnect] section of WIN.INI or, if that does not exist, the directory from which the ICS Communications Manager DLL is executing. See the ICS Installation and Configuration Guide for more information.

IC_DATADIR
This is the directory type for retrieving the name of the directory that contains INFOConnect data files. For standalone and publish installations, this directory name is either the DataDir entry from the [INFOConnect] section of WIN.INI or, if that does not exist, the Windows Directory. For other types of installation, this directory name is retrieved from the IcMgr.INI file. See the ICS Installation and Configuration Guide for more information.

IC_MASTERDIR
This is the directory type for retrieving the name of the directory that contains INFOConnect master database. The master database contains the administrative configurations created during network installation of INFOConnect.

IC_MGR_INI
This is the type used for retrieving the fully qualified filename of the IcMgr.INI file that is currently in use. The IcMgr.INI file is used to record directory-type information and installation options for Local standalone and
LAN installations. See the *ICS Installation and Configuration Guide* for more information.

---

**IC_EMU_LEVEL**

The Emergency Maintenance Upgrade level. This identifier appears after the **IC_MINOR_VERSION** in the version information string. It is an alphabetic identifier, A through Z (mapping 1 through 26) that distinguishes emergency upgrades that occur between minor software releases. For non-emergency release levels, this is zero and does not appear in the version string, **IC_VERSION_STRING**.

---

**IC_ERROR_INFO**

The ICS informative result, or error, type. An error in the range of this type indicates that the request succeeded and suggests that the application log this minor error (or information) for future reference, if desired. It need not be displayed to the user.

**Note:** *ICS DosLink applications cannot use IcDefaultErrorProc to display errors. As of the current INFOConnect release level, the application cannot use IcGetString to retrieve the error string. A future release will allow the use of IcGetString so that the application can display the error string itself.*

---
ICS Data Structures/Types

- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IcDefaultErrorProc function
- ic_default_error_proc function
- IC_ERROR_MASK data type

IC_ERROR_MASK

The mask used to determine the range of the type of the error result. See the IDK Developer's Guide for an example of using this mask.

- WIN
- XVT
- DosLink

See also:
- IC_CHECK_RESULT_SEVERE macro

IC_ERROR_SEVERE

The ICS serious error type indicating that this particular request failed. An error in the range of this type must be displayed to the user.

**Note:** ICS DosLink applications cannot use IcDefaultErrorProc to display errors. As of the current INFOConnect release level, the application cannot use IcGetString to retrieve the error string. A future release will allow the use of IcGetString so that the application can display the error string itself.

- WIN
- XVT
- DosLink

See also:
- IcDefaultErrorProc function
- ic_default_error_proc function
The ICS fatal error type indicating that this request failed. All other requests using the associated session handle will also fail. Therefore, an error in the range of this type suggests that the communication session be closed. If the default error procedure is called, the error message will be displayed to the user and the communication session will be closed automatically.

Note: ICS DosLink applications cannot use IcDefaultErrorProc to display errors. As of the current INFOConnect release level, the application cannot use IcGetString to retrieve the error string. A future release will allow the use of IcGetString so that the application can display the error string itself.

See also:
- IcDefaultErrorProc function
- ic_default_error_proc function
- IC_ERROR_MASK data type

Appendix C for a list of ICS errors

The ICS warning result, or error, type. An error in the range of this type indicates that the request succeeded and suggests that the error should either be displayed to the user or logged by the application for future reference. User intervention (for example, re-configuring or upgrading the software) will remove the warning of this type. The default error procedure will display errors of this type.

Note: ICS DosLink applications cannot use IcDefaultErrorProc to display errors. As of the current INFOConnect release level, the application cannot use IcGetString to retrieve the error string. A future release will allow the use of IcGetString so that the application can display the error string itself.

● WIN ● XVT ● DosLink

● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL
ICS Data Structures/Types

See also:
ICS Default Error Proc function
ic_default_error_proc function
IC_ERROR_MASK data type

Appendix C for a list of ICS errors

IC_FIELD_FLAGS
Data dictionary field flags.

IC_FF_ALTERNATE_KEY
Data dictionary field flag that marks the field as being an alternate key.

IC_FF_LINK_KEY
Data dictionary field flag that marks the field as having the same field name and field type as the primary key in another table.

IC_FF_LINK_KEY_CHANNEL
This data dictionary field flag marks a field in a primary key structure as a link to channel information. This field must be at least IC_MAXCHANNELIDSIZE big. The structure should also include a field for the unique portion of the primary key. This allows the IcAdmin utility to locate and automatically copy the channel related information that exists in a library's invisible table.
For example, the primary key in a library's invisible table that contains channel-related information would be declared as follows:

```
ID_CHAN_HOSTDATA, IC_FF_PRIMARY_KEY, IC_FT_STRUCTURE, 0, 256
ID_CHANNEL, IC_FF_LINK_KEY_CHANNEL, IC_FT_STRINGI, 0, 128
ID_HOSTDATA, IC_FF_NO_KEY, IC_FT_STRINGI, 128, 128
/* define the remaining fields here */
```

**IC_FF_LINK_KEY_PATH**

This data dictionary field flag marks a field in a primary key structure as a link to path information. This field must be at least \texttt{IC_MAXPATHIDSIZE} big. The structure should also include a field for the unique portion of the primary key. This allows the \texttt{IcAdmin} utility to locate and automatically copy the path related information that exists in a library's invisible table.

For example, the primary key in a library's invisible table that contains path-related information would be declared as follows:

```
ID_PATH_WSDATA, IC_FF_PRIMARY_KEY, IC_FT_STRUCTURE, 0, 256
ID_PATH, IC_FF_LINK_KEY_CHANNEL, IC_FT_STRINGI, 0, 128
ID_WSDATA, IC_FF_NO_KEY, IC_FT_STRINGI, 128, 128
/* define the remaining fields here */
```

**IC_FF_NO_KEY**

Data dictionary field flag that is used for fields that are not keys.

**IC_FF_PRIMARY_KEY**

Data dictionary field flag that marks a field as a unique key. The \texttt{IC_TF_PATHTABLE} and \texttt{IC_TF_CHANNELTABLE} table keys are managed by the ICS Manager. They should not have primary key fields. \texttt{IC_TF_INVISIBLETABLE} tables are managed directly by the library. These tables must have one primary key field and it must be the first field of the table.
ICS Data Structures/Types

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

IC_DICT_FIELD data structure

IC_FIELDTYPE

These ICS flags are used in a library's data dictionary to mark the field type.

IC_FT_BINARY

Data dictionary field flag that marks the field as type binary.

IC_FT_BOOL

Data dictionary field flag that marks the field as type boolean.

IC_FT_CHAR

Data dictionary field flag that marks the field as a case sensitive, character type field that is not necessarily null-terminated. If the input data length is shorter, the field will be padded to length with null characters.

IC_FT_INT

Data dictionary field flag that marks the field as type integer.

IC_FT_STRING

Data dictionary field flag that marks the field type as a null-terminated string that is case sensitive.

IC_FT_STRINGI

Data dictionary field flag that marks the field type as a case insensitive, null-terminated string. Note that stringi does not refer to the string form of integers.

IC_FT_STRUCTURE

Data dictionary field flag that marks the fields following as part of the given structure. The fields included in the structure are those that fall within the structure's BitOffset and (BitOffset + BitLength).
IC_FT_UNSIGNED
Data dictionary field flag that marks the field as type unsigned integer.

IC_FST_COMPONENT
The subtype field flag that marks the field as type IC_COMPONENT.

IC_FST_COUNTER
The subtype field flag that marks the field as a counter.

IC_FST_GAUGE
The subtype field flag that marks the field as a counter for a gauge.

IC_FST_ICVER
The subtype field flag that marks the field as type IC_VER.

IC_FST_REVISIONNUM
The subtype field flag that marks the field as type IC_REVISIONNUM.

IC_FST_SERIALNUM
The subtype field flag that marks the field as type IC_SERIALNUM.

IC_FST_TIMETICK
The subtype field flag that marks the field as a counter for timer ticks.

IC_FTX_COMPONENT
The extended field type (type/subtype composite) tag used in the library RC file to mark the IC_COMPONENT field of an INFOConnect table.

IC_FTX_COUNTER
The extended field type (type/subtype composite) tag used in the library RC file to mark a field of an INFOConnect table as a counter.

IC_FTX_GAUGE
The extended field type (type/subtype composite) tag used in the library RC file to mark a field of an INFOConnect table as a counter for a gauge.
ICS Data Structures/Types

**IC_FTX_ICVER**

The extended field type (type/subtype composite) tag used in the library RC file to mark a field of an INFOConnect table as type IC_VER.

**IC_FTX_REVISIONNUM**

The extended field type (type/subtype composite) tag used in the library RC file to mark a field of an INFOConnect table as type IC_REVISIONNUM.

**IC_FTX_SERIALNUM**

The extended field type (type/subtype composite) tag used in the library RC file to mark a field of an INFOConnect table as type IC_SERIALNUM.

**IC_FTX_TIMETICK**

The extended field type (type/subtype composite) tag used in the library RC file to mark a field of an INFOConnect table as a counter for timer ticks.
ICS Data Structures/Types

See also:

IC_DICT_FIELD data structure

IC_HEADER_SIZE

The size of the header in the INFOConnect RCDATA section of the resource file. This includes the version field up to and including the ConfigRcld field. See the IC_RC_NODE data structure.

See also:

IC_RC_NODE data structure

IC_HEADER_3_0

The size of the header in the INFOConnect RCDATA section of the resource file for the 3.0 Release. This includes the version field up to and including the SupplierNum field. See the IC_RC_NODE data structure.

See also:

IC_RC_NODE data structure
IC_KEY_SERIALNUM

This is the special, reserved key number for referencing the SerialNum key of table records. The SerialNum is the unique serial number for the record.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_LCL_FLAGS

This type designates which kind of local action should be performed.

IC_LCL_CLOSESESSION

Flag used to cancel the pending receive and transmit requests just before a session is closed. This flag may be processed specially by the library by preparing for an impending session close. (The library must not attempt to use any transmit or receive buffers or send any messages for that session while waiting for it to close.) IC_LCL_CLOSESESSION is used in combination with IC_LCL_RCVXMT.

IC_LCL_RCV

This flag is used to cancel the pending request to receive data.

IC_LCL_RCVXMT

Flag used to cancel both the pending receive and pending transmit requests.

IC_LCL_XMT

Used to cancel the pending request to transmit data.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IcLcl function
ie_lcl function
ICS Data Structures/Types

ILcLibLcl function
IlcMgrLcl function

IC_LIBRARY_FLAGS

These library flags are used in the IC_RC_NODE of a library’s resource file.

IC_LF_ERRORHELP

Error flag that is to be included in the IC_RC_NODE resource type for libraries that have context sensitive help topics for every library-defined error value. The help topic number for the text of the error must correspond to the IC_RESULT_VALUE of the error. This is used by INFOConnect Connectivity Services to provide trouble-shooting help from the ICS default error dialog.

● WIN
○ XVT
○ DosLink

○ Accessory
○ Shell
○ Configurator
● AIL
● SL
● EIL

See also:

IC_RC_NODE data structure

IC_MAXACCESSORYIDLEN

The maximum length of an accessory identifier (accessory ID), not including the terminating null character.

● WIN
● XVT
● DosLink

● Accessory
● Shell
● Configurator
● AIL
● SL
● EIL
ICS Data Structures/Types

IC_MAXACCESSORYIDSIZE

The maximum size of an accessory identifier (accessory ID), including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXCHANNELIDLEN

The maximum length of a channel identifier (channel ID), not including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXCHANNELIDSIZE

The maximum size of a channel identifier (channel ID), including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXCONNECTEDPATHIDLEN

The maximum length of a dynamically connected path identification (path ID) string, not including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
IC_MAXDESCRIPTIONSIZE

The maximum size of a description string, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXERRORINSERT

The maximum size of a string inserted into an error string, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IcSetSessionError function
IcLibGetString function

IC_MAXERRORSTRING

The maximum length of an error string, not including the terminating null character. Every library-specific error must have an associated string for displaying the error to the user. If the library uses the IcSetSessionError utility, the string may contain up to three string inserts (%s formatting ONLY).

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
See also:

- IcSetSessionError function
- IcLibGetString function

IC_MAXFILENAME SIZE

The maximum size of a fully qualified filename, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXIDSIZE

The maximum size of a library or accessory identifier (or key), including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXLIBRARYIDLEN

The maximum length of a library identifier (library ID), not including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
IC_MAXLIBRARYIDSIZE
The maximum size of a library identifier (library ID), including the terminating null character.

- WIN
- XVT
- Accessory
- AIL
- Shell
- SL
- Configurator
- EIL

IC_MAXPACKAGEIDSIZE
The maximum size of a package ID, including the terminating null character.

- WIN
- XVT
- Accessory
- AIL
- Shell
- SL
- Configurator
- EIL

IC_MAXPATHIDLEN
The maximum length of a path identification (path ID) string, not including the terminating null character.

- WIN
- XVT
- Accessory
- AIL
- Shell
- SL
- Configurator
- EIL

IC_MAXPATHIDSIZE
The maximum size of a path identifier (path ID), including the terminating null character.

- WIN
- XVT
- Accessory
- AIL
- Shell
- SL
- Configurator
- EIL
ICS Data Structures/Types

IC_MAXPRINTSTRING
The maximum size of a printable string. That is, the size of print buffer parameter of IcLibPrintConfig, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IcLibPrintConfig function

IC_MAXSESSIONIDLEN
The maximum length of a communication session identification (session ID) string, not including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXSESSIONIDSIZE
The maximum size of a communication session identification (session ID) string, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
**IC_MAXSESSIONIDSUFFIX**

The maximum number of bytes that are returned from IcLibIdentifySession that are used in creating the session ID string.

- WIN
- Accessory
- AIL
- XVT
- DosLink
- Shell
- SL
- Configurator
- EIL

**IC_MAXSTRINGLENGTH**

The maximum length of a string, not including the terminating null character.

- WIN
- Accessory
- AIL
- XVT
- DosLink
- Shell
- SL
- Configurator
- EIL

**IC_MAXTEMPLATEIDLEN**

The maximum length of a template identifier (template ID), not including the terminating null character.

- WIN
- Accessory
- AIL
- XVT
- DosLink
- Shell
- SL
- Configurator
- EIL

**IC_MAXTEMPLATEIDSIZE**

The maximum size of a template identifier (template ID), including the terminating null character.

- WIN
- Accessory
- AIL
- XVT
- DosLink
- Shell
- SL
- Configurator
- EIL
ICS Data Structures/Types

IC_MAXVENDORNAMELEN
The maximum length of a vendor name, not including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXVENDORNAMESIZE
The maximum size of a vendor name, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MAXWSIDSIZE
The maximum size of a workstation ID, including the terminating null character.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MEMHND
INFOConnect Connectivity Services global buffer handle type for non-shared, intra-application memory.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
IC_MINOR_VERSION

The minor portion of the version. This number appears after the '.' in the version string, IC_VERSION_STRING.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_MSG_CONFIG

These are the indices for the Configuration Accessory API messages that are generated when a configuration data table has been altered. These messages must have been previously registered by the accessory before they can be sent to the accessory.

The messages are sent to all configurators that have the altered configuration object open.

IC_ADD_CONFIG

The index for the Configuration Accessory API message that is generated when a configuration data record has been added to a configuration table.

If the initial attempt to post this message fails, posting the message will be retried after some time interval until either it succeeds, or until a message is available for a configuration record with another serial number. In this case, all configuration messages will queue up until an IC_REFRESH_CONFIG is successfully delivered.

If an IC_ADD_CONFIG message has not yet been delivered and an IC_DELETE_CONFIG message for the same configuration record (that is, a configuration record with the same serial number) becomes available, no message is delivered.

If an IC_ADD_CONFIG message has not yet been delivered and an IC_UPDATE_CONFIG message for the same configuration record becomes available, only the IC_ADD_CONFIG message is delivered.

This message is message index 1.

IC_DELETE_CONFIG

This is the index for the Configuration Accessory API message that is generated when a configuration data record has been deleted from a configuration table.
ICS Data Structures/Types

If the initial attempt to post this message fails, posting the message will be retried after some time interval until either it succeeds, or until a message is available for a configuration record with another serial number. In this case, all configuration messages will queue up until an IC_REFRESH_CONFIG is successfully delivered.

It is message index 3.

**IC_REFRESH_CONFIG**

This message is posted initially. It is also posted after some time interval when an attempt to post an IC_ADD_CONFIG, IC_UPDATE_CONFIG, or IC_DELETE_CONFIG message for multiple configuration records (that is, configuration messages with different serial numbers) fails.

This message is message index 0.

**IC_UPDATE_CONFIG**

This is the index for the Configuration Accessory API message that is generated when a configuration data record has been updated in a configuration table.

If the initial attempt to post this message fails, posting the message will be retried after some time interval until either it succeeds, or until a message is available for a configuration record with another serial number. In this case, all configuration messages will queue up until an IC_REFRESH_CONFIG is successfully delivered.

If an IC_UPDATE_CONFIG message has not yet been delivered and an IC_DELETE_CONFIG message for the same configuration record (that is, a configuration record with the same serial number) becomes available, only the IC_DELETE_CONFIG message is delivered.

It is message index 2.
IC_NEXTEVENT_FLAGS

These flags are used by ICS DosLink applications that use a callback function (in contrast to polling using `IcGetNextEvent`).

**IC_NEXTEVENT_CHECK**

Used to check the message queue for messages.

**IC_NEXTEVENT_POP**

Used to request that the current message be popped from the message queue.

**IC_NEXTEVENT_READY**

Flag used to inform ICS that the callback routine is ready to receive the next message. ICS DosLink applications must follow each call to all ICS APIs with a call to `IcNextEvent` with this flag.

**IC_NEXTEVENT_TIMER**

Used to set a timer value.
ICS Data Structures/Types

IC_OK

An IC_RESULT indicating no error.

IC_OPEN_OPTIONS

ICS option flags that indicate the open options for IcLibOpenChannel and IcLibOpenSession.

IC_OPEN_VERIFY

If (Options & IC_OPEN_VERIFY) is true on the IcLibOpen... function call, the library should return IC_VERIFY_OK if an attempt to open this session would succeed. It should return an error if an attempt to open this session would fail. This information is used to prune the list of path IDs available to the user from the select path dialog.
See also:

IcLibOpenSession function

IC_PACKAGE

Package identification that defines an ID as a package ID.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_PATH_FLAGS

ICS flags that indicate the status of a path.

IC_PF_HIDDEN

ICS flag indicating that a path is hidden. That is, the path will not appear as a choice in the ICS Select Path dialog box.

IC_PF_SYSTEM

ICS flag indicating a library ID.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_PRINT_SUMMARY

A flag that requests summary format of the library's configuration data.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
ICS Data Structures/Types

See also:

IkLibPrintConfig function

IC_RC_NODE
typedef struct aICRCNode {
    unsigned version;
    unsigned revision;
    IC_COMPONENT_TYPE type;
    unsigned header_size;
    unsigned dictionary_id;
    unsigned id;
    unsigned description;
    unsigned vendor;
    unsigned module_id;
    ICSESSION_FLAGS session_flags;
    IC_LIBRARY_FLAGS library_flags;
    unsigned ConfigRcId;
    /* The following fields were added in the 2.02 Release */
    /* Use the IC_HEADER_3_0 for this resource header size */
    unsigned max_version;
    unsigned max_revision;
    unsigned subtype;
    unsigned GenericDictMap;
    IC_COMPONENT GenericNum
    IC_COMPONENT SupplierNum
} IC_RC_NODE;

This data structure type defines the format of the INFOConnect RCDATA resource.
INFOConnect is defined in the icdef.h include file. This resource is required for all
INFOConnect Connectivity Services libraries and accessories. Refer to Microsoft®
Windows™ Software Development Kit Reference, User-Defined Resource Statement
section for more information.

version The oldest level of Connectivity Services that
this component supports. See
IC_VERSION_

revision The oldest level of Connectivity Services that
this component supports. See
IC_REVISION_
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The <strong>IC_COMPONENT_TYPE</strong> of the file.</td>
</tr>
<tr>
<td>header_size</td>
<td>The size of the header. See <strong>IC_HEADER_</strong>....</td>
</tr>
<tr>
<td>dictionary_id</td>
<td>The numeric ID of the dictionary resource RCDATA, formatted as an <strong>IC_DICT_NODE</strong>, or zero if no data dictionary tables exist.</td>
</tr>
<tr>
<td>id</td>
<td>The string resource string number of the file's ID. See Appendix A for a list of ICS Standard IDs. This ID is used as the default when the library or accessory is installed.</td>
</tr>
<tr>
<td>description</td>
<td>The string resource string number of the file's description. This description is used as the default when the library or accessory is installed.</td>
</tr>
<tr>
<td>vendor</td>
<td>The string resource string number of the vendor identification.</td>
</tr>
<tr>
<td>module_id</td>
<td>The string resource string number of the file's module ID. This should be the name from the .DEF file and may be the same as id.</td>
</tr>
<tr>
<td>session_flags</td>
<td>The pertinent <strong>IC_SESSION_FLAGS</strong> or zero if not applicable.</td>
</tr>
<tr>
<td>library_flags</td>
<td>The pertinent <strong>IC_LIBRARY_FLAGS</strong> or zero if not applicable.</td>
</tr>
<tr>
<td>ConfigRcId</td>
<td>The numeric ID of the template configuration resource RCDATA (<strong>IC_Template</strong>...), or zero if one doesn't exist.</td>
</tr>
<tr>
<td>max_version</td>
<td>The highest level of Connectivity Services that this component supports. See <strong>IC_VERSION_</strong>....</td>
</tr>
<tr>
<td>max_revision</td>
<td>The highest level of Connectivity Services that this component supports. See <strong>IC_REVISION_</strong>....</td>
</tr>
<tr>
<td>subtype</td>
<td>Reserved. Must be zero.</td>
</tr>
<tr>
<td>GenericDictMap</td>
<td>If this is a trace hook library, this is the ordinal value for the <strong>IcLibTrace</strong> entry point. Otherwise, this must be zero.</td>
</tr>
</tbody>
</table>
ICS Data Structures/Types

GenericNum
The generic component value. Note that this LONG value must appear in the RC file as two values: LO, HI.

SupplierNum
The supplier component value. Note that this LONG value must appear in the RC file as two values: LO, HI.

Note: During installation, the library/accessory ID and description will be extracted from the string table and used as defaults when adding the library/accessory to the ICS database. The CONFIGRCID RCDATA is only valid for accessories and libraries. It will be parsed and the resulting templates added to the ICS database. Refer to Microsoft® Windows™ Software Development Kit Reference, User-Defined Resource Statement section for more information. See IC_TemplateInit for an example of the format of the CONFIGRCID RCDATA resource.

● WIN  ● XVT  ● DosLink

● Accessory  ● Shell  ● Configurator

● AIL  ● SL  ● EIL
See also:

IC_DICT_NODE  data type
IC_HEADER_SIZE  data type
IC_ACCESSORY  data type
IC_SERVICE  data type
IC_INTERFACE  data type
IC_SESSION_FLAGS  data type
IC_LIBRARY_FLAGS  data type
IC_Template...  data types

**IC_RECORD_INFO**

typedef struct {
  IC_SERIALNUM SerialNum;
  IC_REVISIONNUM RevisionNum;
} IC_RECORD_INFO;

This data structure type defines the informational fields for the records of the active path (session) and active channel tables.

*SerialNum*  The unique serial number for the record.

*RevisionNum*  The count of the number of times this record has been modified.

● WIN  ● XVT  ○ DosLink

○ Accessory  ○ Shell  ● Configurator
○ AIL  ○ SL  ○ EIL
ICS Data Structures/Types

**IC_RECORD_SIZE**

The size of an IC_DICT_FIELD data type in bytes.

- **WIN**
- **XVT**
- **DosLink**
- **Accessory**
- **Shell**
- **Configurator**
- **AIL**
- **SL**
- **EIL**

See also:

**IC_DICT_FIELD** data type

**IC_RESULT**

INFOConnect Connectivity Services type used to communicate status and error information. It is the type returned by most INFOConnect functions and by most ICS events. It consists of the following three parts:

**IC_RESULT_CONTEXT**

The context that defines the result.

**IC_RESULT_TYPE**

The result type.

**IC_RESULT_VALUE**

The result value.

*Note:* See Appendix B for a list of ICS statuses. See Appendix C for a list of ICS errors.

- **WIN**
- **XVT**
- **DosLink**
- **Accessory**
- **Shell**
- **Configurator**
- **AIL**
- **SL**
- **EIL**
See also:

IC_ERROR_MASK data type
IC_ERROR_INFO data type
IC_ERROR_WARNING data type
IC_ERROR_SEVERE data type
IC_ERROR_TERMINATE data type
IC_RESULT_SUBTYPE data type
IC_RESULT_SUBVALUE data type

IC_RESULT_CONTEXT_CFG
The INFOConnect Configuration Accessory result context.
● WIN ● XVT ● DosLink
● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL

IC_RESULT_CONTEXT_ICDB
The result context of the INFOConnect Database DLL.
● WIN ● XVT ● DosLink
● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL

IC_RESULT_CONTEXT_ICUTIL
The result context of the INFOConnect Utilities DLL.
● WIN ● XVT ● DosLink
● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL
ICS Data Structures/Types

IC_RESULT_CONTEXT_INVALID
The INFOConnect invalid result context.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

IC_RESULT_CONTEXT_STD
The INFOConnect standard result context for the ICS Manager.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

IC_RESULT_SUBTYPE
INFOConnect Connectivity Services type that interprets part of
IC_RESULT_VALUE as a subtype field. This is used by the IC_STATUS_UTS
status message. See Appendix B for more information on this status message.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

- IC_RESULT_VALUE data type
- IC_RESULT_SUBVALUE data type
IC_RESULT_SUBVALUE

INFOConnect Connectivity Services type that interprets part of IC_RESULT_VALUE as a subvalue field. This is especially useful for the IC_STATUS_UTS status message. See Appendix B for more information on this status message.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_RESULT_VALUE data type
IC_RESULT_SUBTYPE data type

IC_REVISION_...

Used to delimit the range of release revisions of the ICS API supported by this component.

IC_REVISION_1_0
    ICS Revision 1.0.

IC_REVISION_1_2
    ICS Revision 1.2.

IC_REVISION_2_0
    ICS Revision 2.0.

IC_REVISION_2_02
    ICS Revision 2.02.

IC_REVISION_3_0
    ICS Revision 3.0.

Etc.
IC_RevisionNum

ICS data type for the count of the number of times a configuration record has been modified.

See also:

IC_Record_Info data structure

IC_SerialNum

ICS data type for the unique serial number of a configuration record.

See also:

IC_Record_Info data structure
IC_SESSION_FLAGS

These session flags are used in the IC_RC_NODE of a library's resource file.

IC_SF_SESSIONSTATUS

This flag is to be included in the IC_RC_NODE resource type for libraries that can respond to the IC_CONNECT_STATUS status type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IC_RC_NODE data structure
- IC_STATUS_CONNECT status type
IC_SINFO

typedef struct aSINFO {
    long max_size;
    unsigned transparent:1;
    unsigned block_mode:1;
    unsigned reliable:1;
    unsigned focus_notify:1;
    unsigned server:1
    unsigned untrans8:1
    unsigned connect:1
    unsigned reconnect:1
    unsigned autoreconnect:1
    unsigned :1
    unsigned :1
    unsigned :1
    unsigned :1
    unsigned :1
    unsigned :1
    short padword;
    long padlong[6];
} IC_SINFO;

This data structure type defines a record of pertinent information about a communication session.

*max_size*  
The maximum size of a transmission block or a received block of data that can be transported across the connection.

*transparent*  
Signifies whether or not all binary data streams can be safely sent across this connection. FALSE implies that, at the very least, only the alpha-numerics, period, comma, and ESC can be safely sent.

*block_mode*  
Signifies whether or not the data is sent and received as messages or just as a stream of characters. For some libraries, the **IC_STATUS_BLOCKING** status can alter the state of this field.
**reliable**
Signifies whether or not undelivered messages are signaled to the application (usually through communication session failure).

**focus_notify**
Signifies whether or not the application should call the Set Status procedure with **IC_REACTIVATE_ON** or **IC_REACTIVATE_OFF** each time it gains or loses focus.

**server**
Signifies whether or not this is a server.

**untrans8**
Reserved for future use.

**connect**
Signifies whether or not the library generates **IC_CONNECT_OPEN** and **IC_CONNECT_CLOSE** statuses.

**reconnect**
Signifies whether or not the library honors **IC_CONNECT_OPEN**, **IC_CONNECT_CLOSE**, and **IC_CONNECT_EOF** statuses from the application.

**autoreconnect**
Signifies whether or not the library will assume an **IC_CONNECT_OPEN** status whenever it generates an **IC_CONNECT_CLOSE** status.

**unsigned**
Reserved for future use.

**padword**
Reserved for future use.

**padlong**
Reserved for future use.
ICS Data Structures/Types

● WIN ● XVT ● DosLink
● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL

See also:

IcGetSessionInfo function
ic_get_session_info function
IcSetStatus function
ic_set_status function
IcLibGetSessionInfo function
IC_STATUS_BLOCKING data type
IC_STATUS_REACTIVATE data type

IC_STATUSBUF
typedef struct aSTATUSBUF {
  IC_RESULT icstatus;
  IC_RESULT icerror;
  long reserved;
  unsigned uBufSize;
  unsigned uDataSize;
} IC_STATUSBUF;

This data structure type defines a header for a data buffer that is used with the extended status IC_STATUS_BUFFER.

icstatus The actual status message associated with the data buffer of information.
icerror The IC_RESULT of the status request.
reserved Reserved for future use.
uBufSize The actual size, in bytes, of the data buffer.
uDataSize The size, in bytes, of the valid data.
**Notes:**

- The data buffer must immediately follow the `IC_STATUSBUF` header. It should not contain pointers, but may contain offsets within the structure.

- The `IC_STATUS_BUFFER` status message request can be synchronous or asynchronous. For a synchronous message, the library receives and processes the `IC_STATUS_BUFFER` status message, setting the `icerror` field to either `IC_OK` or an error. The `icerror` result is returned, and appears to the application as an `IC_STATUSRESULT` message. An `IC_OK` result implies that the data buffer has been accessed and `uDataSize` is the size of the valid information.

For an asynchronous message, the library receives the `IC_STATUS_BUFFER` status message, sets the `icerror` field to `IC_INCOMPLETE` and returns `IC_INCOMPLETE`. The application receives the `IC_StatusResult` message with the `IC_INCOMPLETE` result. When the library finally supplies the `uDataSize` and accesses the data buffer, the `icerror` field should be set to `IC_COMPLETE` or to an error and the `IC_STATUS_BUFFER` status message should be sent back to the application.

In both cases, the application should be responsible for freeing the buffer. The buffer should not be freed, however, until after the application receives an `IC_StatusResult` response of `IC_OK` or an error. If the result is `IC_INCOMPLETE`, this is the asynchronous case and the buffer should not be freed until the status is returned via the `IC_STATUS_BUFFER` status message.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

`IC_STATUS_BUFFER` status type

**IC_STATUS_BLOCKING**

This is an application-initiated status type that is used to control the blocking mode of those service libraries that support it.

**IC_BLOCKING_ON**

Turn blocking mode ON.
### ICS Data Structures/Types

**IC_BLOCKING_OFF**

Turn blocking mode OFF.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL
See also:

- IcSetStatus function
- ic_set_status function
- IcLibSetResult function
- IC_SINFO data structure

**IC_STATUS_BUFFER**

IC_STATUS_BUFFER is an extended status that allows a buffer of information to be exchanged between ICS components. It is to be used whenever the status information to be exchanged exceeds the limits of the IC_RESULT structure. In this case, the IC_RESULT_VALUE portion of the status message is an INFOConnect buffer handle of type HIC_STATUSBUF.

The IC_STATUS_BUFFER status message request can be synchronous or asynchronous. For a synchronous message, the library receives and processes the IC_STATUS_BUFFER status message, setting the icerror field to either IC_OK or an error. The icerror result is returned to the application as an IC_STATUSRESULT message. An IC_OK result implies that data has been accessed and uDataSize is the size of the valid information.

For an asynchronous message, the library receives the IC_STATUS_BUFFER status message, sets the icerror field to IC_INCOMPLETE and returns IC_INCOMPLETE. The application receives the IC_STATUSRESULT message with the IC_INCOMPLETE result. When the library finally supplies the uDataSize and data, the icerror field should be set to IC_COMPLETE or to an error and the IC_STATUS_BUFFER status message should be sent back to the application using IcMgrSendEvent.

In both cases, the application should be responsible for freeing the buffer. The buffer should not be freed, however, until after the application receives an IC_STATUSRESULT response of IC_OK or an error. If the result is IC_INCOMPLETE, this is the asynchronous case and the buffer should not be freed until the status is returned via the IC_STATUS_BUFFER status message.
ICS Data Structures/Types

● WIN  ● XVT  ● DosLink

● Accessory  ● Shell  ● Configurator
● AIL  ● SL  ● EIL

See also:
  IcSetStatus function
  ic_set_status function
  IcLibSetResult function

IC_STATUS_COMMMGR

This status type originates from the ICS Manager itself and conveys initialization or termination information. If the ICS Manager terminates, the ICS accessory must call the ICS initialization routine before calling any other ICS procedures.

IC_COMMMGR_CANCELEXIT

Status sent to all ICS communications sessions that previously received an IC_COMMMGR_QUERYEXIT status when one of the applications calls IcExitOk(FALSE).

IC_COMMMGR_EXIT

Status sent to all ICS communications sessions if IcExitOk(FALSE) is never called. The ICS Manager will then exit.

IC_COMMMGR_INITIALIZED

Status sent to all Windows applications when the ICS Manager finishes initializing. ICS accessories may now call the ICS initialization routine, if necessary, before establishing INFOConnect sessions.

IC_COMMMGR_QUERYEXIT

Status sent to all ICS communications sessions when the user closes the INFOConnect Shell. If the application does not wish to close the session, it should cancel the exit by calling IcExitOk(FALSE).
**IC_COMMMGR_QUERYSHUTDOWN**

Status sent to all ICS communications sessions when Windows is exiting. If the application does not wish to close the session, it should cancel the exit by calling `IcExitOk(FALSE)`.

**IC_COMMMGR_REINSTALL**

Status posted to all windows by `install.exe` when the ICS Manager is being reinstalled.

**IC_COMMMGR_TERMINATED**

Status sent to all Windows applications when the ICS Manager finishes terminating. All ICS accessories should either close or call the ICS initialization routine before establishing another ICS session.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**IC_STATUS_CONNECT**

When initiated from an external interface library, this status type signifies the state of the connection. When initiated from an application, a networking external interface library is instructed to alter the state of the connection, if possible. The connection states are defined by one of the following statuses.

**IC_CONNECT_ACTIVITY**

The physical connection (not necessarily this communication session) is functioning as expected.

**IC_CONNECT_CLOSE**

The logical connection is NOT available for bi-directional communication under the current configuration.

**IC_CONNECT_EOF**

The logical communication session is physically closed (for use under TCP/IP).

**IC_CONNECT_NOACTIVITY**

The physical connection is NOT functioning as expected.
ICS Data Structures/Types

IC_CONNECT_OPEN
The logical connection is available for bi-directional communication under the current configuration.

IC_CONNECT_STATUS
Status originating from the application requesting that the EIL display status information to the user. This is the status sent when the user selects the Status button from the INFOConnect Shell.

IC_CONNECT_BROKEN
Status that indicates that the other half of two connected sessions has closed.

IC_CONNECT_JOINED
Status that indicates that two sessions have been connected. For example, this is the status received when two DosLink sessions are connected.

IC_CONNECT_SERVER
Status that originates from the server application (such as the DosLink Server accessory) that indicates readiness to the client.

Note: Library developers should take care not to generate an over abundance of status messages to prevent thrashing. This is especially important on entry level workstations that may have insufficient memory to execute the current application mix.

● WIN ● XVT ● DosLink
● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL

See also:
IcSetStatus function
ic_set_status function
IcLibSetResult function
IcSetServerInfo function

IC_STATUS_CONTROL
When initiated from an external interface library, this status type signifies that a request is being made to the application. When initiated from an application, it
signifies a request to another connected application. The requests are defined by one of the following statuses.

**IC_CONTROL_ACTIVATE**

This status requests that the application's window become active for user input. It is usually initiated from the ICS Shell through the GoTo button.

**IC_CONTROL_RCVREADY**

This status requests that the application perform a receive request if it does not already have a request outstanding.

**IC_CONTROL_RCVAVAIL**

This is a notification that a message is available but not deliverable due to the state of the application.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IcSetStatus
- ic_set_status
- IcLibSetResult
- IcOpenAccessory
- ic_open_accessory
IC_STATUS_DATAFLAGS

This application-initiated status controls the state of messages. It is used with the IcSetStatus/IcXmt functions and the IC_RCVDONE event to mark expedited and/or multipart messages. The initial state of messages is assumed to be IC_DATAFLAGS_NONE.

IC_DATAFLAGS(v)

This is a macro that creates an IC_DATAFLAGS status with status value v.

IC_DATAFLAGS_EXPEDITED

This status indicates that the following transmitted message is urgent. It is to be delivered ahead of any other messages in the message queue.

IC_DATAFLAGS_MORE

This status indicates that the following transmitted messages are part of a multipart message.

IC_DATAFLAGS_NONE

This status indicates that none of the data flags are set. It is used to indicate the last part of a multipart message. Note that this status is sent before the final part of the message is transmitted.

IC_DATAFLAGS_RESERVED1

Reserved status.

IC_DATAFLAGS_RESERVED2

Reserved status.

Notes:

- The following is an example of sending two expedited messages:

```c
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_EXPEDITED));
IcXmt(); /* send an expedited message */
IcXmt(); /* send another expedited message */
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_NONE)); /* restore default state */
```
The following is an example of sending multipart messages:

```c
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_MORE));
IcXmt(); /* send first part of multipart message */
...
IcXmt(); /* ... */
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_NONE)); /* restore default state */
IcXmt(); /* send last part of multipart message */
...
```

The following is an example of sending multipart, expedited messages:

```c
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_EXPEDITED|IC_DATAFLAGS_MORE));
IcXmt(); /* send first part of expedited message */
...
IcXmt(); /* send middle of expedited message */
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_EXPEDITED));
IcXmt(); /* send last part of expedited message */
...
IcSetStatus(hSession, IC_DATAFLAGS(IC_DATAFLAGS_NONE)); /* restore default state */
IcXmt(); /* send normal message */
...
```

The following shows a portion of the `IC_RCVDONE` case:

```c
IC_RESULT_VALUE dataflags;

case IC_Status:
    if (IC_CHECK_DATAFLAGS(status))
        dataflags = IC_GET_RESULT_VALUE(status);
    break;

case IC_RcvDone:
    if (dataflags & IC_DATAFLAGS_EXPEDITED) {
        /* handle expedited case */
    }
    if (dataflags & IC_DATAFLAGS_MORE) {
        /* handle multipart case */
    }
```
IC_STATUS_FKEY

This application-initiated status type is used to send function key messages to the underlying layers of the ICS communication session. The function keys are defined by one of the following statuses.

IC_FKEY_BREAK

The break key.

IC_FKEY_1

Function key 1.

IC_FKEY_2

Function key 2.

...

...

IC_FKEY_23

Function key 23.

IC_FKEY_24

Function key 24.

IC_FKEY_MSGWAIT

Uniscope-specific break key.

IC_FKEY_SYSMODE

Uniscope-specific OS3 system mode key.
IC_FKEY_WSMODE

Uniscope-specific OS3 workstation mode key.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

IC_STATUS_LINESTATE

This external interface library-initiated status type signifies the state of the underlying layer of the ICS communication session. The line states are defined by one of the following statuses.

IC_LINESTATE_LCL

The ICS communication session is neither transmitting nor receiving.

IC_LINESTATE_RCV

The ICS communication session is in receive mode.

IC_LINESTATE_XMT

The ICS communication session is in transmit mode.

Note: For performance reasons, the DosLink Server filters out the IC_STATUS_LINESTATE status types.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

- IC_STATUS message
- E_IC_STATUS event
- IcSetStatus function
- ic_set_status function
- IcLibSetResult function
IC_STATUS_REACTIVATE

This application-initiated status type is used to notify the underlying communication session layers that the application window has received or lost focus.

IC_REACTIVATE_ON

Application has received focus and $\text{sinfo.focus.notify}$ is TRUE.

IC_REACTIVATE_OFF

Application has lost focus and $\text{sinfo.focus.notify}$ is TRUE.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- IcGetSessionInfo function
- ic_get_session_info function
- IcSetStatus function
- ic_set_status function
- IcLibSetResult function
- IcLibGetSessionInfo function
- IC_SINFO data type
**IC_STATUS_TRANS**

This application-initiated status type is used to notify the backplane of the beginning and end of transactions. The initial state of all applications is assumed to be **IC_TRANSACTION_OFF**.

**IC_TRANSACTION_ON**

Indicates that **IC_TRANSACTION_BEGIN** and **IC_TRANSACTION_END** status will be sent.

**IC_TRANSACTION_OFF**

Indicates that **IC_TRANSACTION_BEGIN** and **IC_TRANSACTION_END** status will not be sent.

**IC_TRANSACTION_BEGIN**

Sent at the beginning of a transaction.

**IC_TRANSACTION_END**

Sent at the end of a transaction.

- **WIN**, **XVT**, **DosLink**, **Accessory**, **Shell**, **Configurator**, **AIL**, **SL**, **EIL**

See also:

- IcSetStatus function
- ic_set_status function
- IcLibSetResult function
IC_STATUS_UTS

This status type may be used to send and receive special messages to/from the UTS external interface libraries. See Appendix B for more information on the special values that this status message supports.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IcGetSessionInfo function
- ic_get_session_info function
- IcSetStatus function
- ic_set_status function
- IC_STATUS message
- E_IC_STATUS event
- IC_RESULT_SUBTYPE data type
- IC_RESULT_SUBVALUE data type

Appendix B

IC_TABLE_FLAGS

These are flags used in a library's data dictionary that mark different kinds of configuration tables.

IC_TF_ACTIVECHANNEL

This flag marks a table as containing channel data for those channels that are involved in active sessions. IC_TF_ACTIVECHANNEL tables must be managed by the library.

By default, the first field of this table is the ICSTD_ACTIVECHANNEL channel ID field.

IC_TF_ACTIVECUSTOM

Flag that marks a table as containing dynamic, custom information. Data in IC_TF_ACTIVECUSTOM tables are managed by the library.
IC_TF_ACTIVEPATH

Data dictionary table flag that marks a table as containing path-type information for active paths. Also called a session table. IC_TF_ACTIVEPATH tables must be managed by the library. The ICS Manager gets access to this information through IcLibAccessConfig.

By default, the first two fields of this table are the ICSTD_ACTIVEPATH path ID field and ICSTD_ACTIVEPATHCHANNEL channel ID field.

IC_TF_CHANNELTABLE

Flag that marks a table as channel data. Data in IC_TF_CHANNELTABLEs is to be made visible to the user through the global configuration dialog. The corresponding database table, along with the table's primary key, is managed by the ICS Manager. There can be zero or one channel tables. If a channel table is defined, a path table must also be defined containing at least one field. The field, however, may be a filler field.

By default, the first field of this table is the ICSTD_CHANNEL channel ID field.

IC_TF_CUSTOMTABLE

Data dictionary table flag that marks a table as being a visible table other than channel or path. This table is managed by the library itself through a dialog box which the user can access through the ICS Shell or Configurator. There can be any number of custom tables.

IC_TF_DYNAMICTABLE

This flag marks a table as containing dynamic data. IC_TF_DYNAMICTABLE tables are invisible and must be managed by the library. The ICS Manager gets access to this information through IcLibAccessConfig.

IC_TF_INVISIBLETABLE

Data dictionary table flag that marks a table as being managed by the library itself. Data in IC_TF_INVISIBLETABLEs must be managed by the library. There can be any number of invisible tables.

IC_TF_PATHTABLE

This flag marks a table as path-specific data. Data in IC_TF_PATHTABLEs is to be made visible to the user through the path configuration dialog. The corresponding database table, along with the table's primary key, is managed by the ICS Manager. There can be zero or one path tables. However, if a
channel table is defined, a path table must also be defined. As of the current INFOConnect release this table must have at least one field. The field, however, may be a filler field.

By default, the first two fields of this table are the ICSTD_PATH path ID field and ICSTD_PATHCHANNEL channel ID field.

**IC_TF_STACKTABLE**
Data dictionary table flag that marks a table as being a stack table.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_DICT_FIELD data structure
- IcLibAccessConfig function
- IcLibUpdateConfig function

**IC_TABLETYPE**
ICS type of configuration tables.

**IC_ACTIVECHANNEL**
Denotes the active channel table.

**IC_ACTIVECUSTOM**
Denotes the active custom table.

**IC_ACTIVEPATH**
Denotes the active path table (also referred to as the session table).

**IC_CHANNEL**
Denotes the channel table.

**IC_CUSTOMTABLE**
Denotes the custom table.

**IC_PATH**
Denotes the path table.
IC-template
Denotes the template table.

IC-unknown
Denotes an unknown table.

See also:
IC-table_flags data type

IC-templateBegin
The ICS template resource flag for starting the definition of a template. This must be followed by the template identification (template ID).

See also:
IC-templateInit data type

IC-templateChannel
The ICS template resource flag for identifying the channel ID in a template definition. This is an optional flag and, if it exists, must be followed by the channel identification.
ICS Template Resource Flag for Identifying the Configuration Library ID of the Configuration Library Associated with This Template. This is an optional flag and, if it exists, must be followed by the library ID of the library that controls configuration for this template. If this flag exists, IC_TemplateConfigTable must also exist.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_TemplateInit data type

IC_TemplateConfigTable data type
IC_TemplateConfigTable

The ICS template resource flag for identifying the table number of the configuration table for the configuration library associated with this template. This is an optional flag and, if it exists, must be followed by the table number of the configuration table as defined in the library's resource file. If this flag exists, IC_TemplateConfig must also exist.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_TemplateInit data type
- IC_TemplateConfig data type

IC_TemplateDescription

The ICS template resource flag for identifying the template description in a template definition. This flag must be followed by the description.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_TemplateInit data type
ICS Data Structures/Types

IC_TemplateEnd
The ICS template resource flag for ending the definition of a template.

- WIN  ○ XVT  ○ DosLink

○ Accessory  ○ Shell  ○ Configurator
● AIL  ● SL  ● EIL

See also:
IC_TemplateInit  data type

IC_TemplateFlags
The ICS template resource flag for identifying the template flags of a template. The template flags, if it exists, can be "H" for hidden (that is, IC_PF_HIDDEN), "S" for system (that is, IC_PF_SYSTEM), or "HS" for hidden and system (that is, IC_PF_HIDDEN & IC_PF_SYSTEM).

- WIN  ○ XVT  ○ DosLink

○ Accessory  ○ Shell  ○ Configurator
● AIL  ● SL  ● EIL

See also:
IC_TemplateInit  data type

IC_TemplateInit
The ICS template resource flag for starting a series of one or more template descriptions.
Notes:

- The following is an example of the format of the `CONFIGRCID RCDATA` resource.

```
CONFIGRCID RCDATA
BEGIN
IC_TemplateInit

IC_TemplateBegin "TP1" /* template id */
IC_TemplateDescription "Trace/Local"
IC_TemplateLibrary "Trace" /* library stack */
IC_TemplateLibrary "Local"
IC_TemplateEnd

IC_TemplateBegin "TP2"
IC_TemplateDescription "Service/Local"
IC_TemplateLibrary "Service"
IC_TemplateChannel "sChanID" /* optional channel ID */
IC_TemplateLibrary "Local"
IC_TemplateEnd

IC_TemplateBegin "TTY"
IC_TemplateDescription "TTY Communications"
IC_TemplateOpenID "ANSI" /* optional OpenID */
IC_TemplateLibrary "TTY"
IC_TemplateEnd

IC_TemplateBegin "Local"
IC_TemplateDescription "Local Communications"
IC_TemplateFlags "H" /* optional flags */
IC_TemplateConfig "Local"
IC_TemplateConfigTable "1002"
IC_TemplateLibrary "Local"
IC_TemplateEnd

IC_TemplateTerm
END
```
ICS Data Structures/Types

There may be any number of template definitions (template definitions are bounded by \texttt{IC_TemplateBegin} and \texttt{IC_TemplateEnd}).

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

\texttt{IC_RC_NODE} data type

\textbf{IC_TemplateLibrary}

The ICS template resource flag for identifying the library ID in a template definition. There may be one or more of these and each must be followed by the library identification. If a channel ID is to be associated with a library in a template, the channel flag must immediately follow the library flag/library ID line in the resource.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

\texttt{IC_TemplateInit} data type

\textbf{IC_TemplateOpenID}

The ICS template resource flag for identifying the template \textit{OpenID} of a template. This flag is optional. If it exists, it must be followed by the template's \textit{OpenID}. The \textit{OpenID} is an identifier, usually a standard accessory ID, with which to associate this template. See Appendix A for a list of standard IDs.
ICS Data Structures/Types

See also:

IC_TemplateInit data type

IC_TemplateTerm

The ICS template resource flag for terminating a list of one or more template definitions.

See also:

IC_TemplateInit data type

IC_UPGRADE_INFO
typedef struct aUPGRADEINFO {
    UINT UpgradeLen;
    UINT OldDataOffset;
    UINT OldDataLen;
    UINT OldDataSerialNum;
    long Reserved2;
    long Reserved3;
} IC_UPGRADE_INFO;

This data structure type defines the data found at (buffer + len) for
IclLibVerifyConfig when Command == IC_VER_UPGRADE.

- **UpgradeLen**: The size, in bytes, of this data structure.
- **OldDataOffset**: The offset of the previously formatted data buffer (or 'old' data) from the start of the IclLibVerifyConfig buffer parameter.
ICS Data Structures/Types

OldDataLen

The size, in bytes, of the old data buffer.

OldDataSerialNum

The serial number of the table from which the old data buffer came.

Reserved2

Reserved for future use.

Reserved3

Reserved for future use.

Note: The following is a pictorial view of the relationship of the new buffer (the buffer parameter to IcLibVerifyConfig), the IC_UPGRADE_INFO data structure, and the old data buffer.

See also: IcLibVerifyConfig function
ICS Data Structures/Types

IC_VER

The ICS type for component versions.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_VER_INFO

typedef union {
  IC_VER IcVer;
  struct {
    WORD Rev;
    WORD Ver;
  } w;
  struct
    BYTE Revision;
    BYTE EmuLevel;
    BYTE MinorVersion;
    BYTE MajorVersion;
  } b;
} IC_VER_INFO;

This data structure type defines the format of the version control information.

IcVer

The ICS IC_VER version number.

Alternate View:

Rev

The Revision portion of IcVer.

Ver

The Version portion of IcVer.
ICS Data Structures/Types

Alternate View:

Revision
The IC_BUILD_REVISION.

EmuLevel
The IC_EMU_LEVEL, 1 through 26 (mapping A through Z), zero if this is not an emergency release.

MinorVersion
The minor, IC_MINOR_VERSION, portion of the version.

MajorVersion
The major portion of the version.

● WIN
● XVT
○ DosLink

● Accessory
● Shell
● Configurator
● AIL
● SL
● EIL

IC_VERIFY
ICS type use to communicate the action to be taken in a library's IcLibVerifyConfig procedure when the configuration is in error. The following commands are defined.

IC_VER_DELETE
The given configuration data is about to be deleted. Perform any special verification/cleanup of related information.

IC_VER_DISPLAY
Verify and display errors to the user.

IC_VER_MODIFY
Verify, displaying errors to the user for modification.

IC_VER_SAVE
The given configuration data is about to be saved. Verify without displaying errors to the user.
IC_VER_UPGRADE

Perform special upgrade processing and data conversions on the given buffer of data. Note that an IC_UPGRADE_INFO data structure is located at \((buffer + len)\) for providing access to the data in the previous format.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IcLibVerifyConfig function
- IC_UPGRADE_INFO data structure

IC_VERIFY_OK

An IC_RESULT indicating that no error occurred during a verify action.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

IC_VERSION_FILE

The default file version for Windows version control.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
- IC_VER_INFO data structure
- icdef.rh include file

IC_VERSION_PRODUCT

The default product version for Windows version control.
ICS Data Structures/Types

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

See also:

IC_VER_INFO data structure
icdef.rh include file

IC_VERSION_STRING

The ICS current version information in string format.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

IC_VERSION...

Used to delimit the range of release levels of the ICS API supported by this component. These are used as the version (oldest version) and the max_version (newest version) values in the INFOConnect RCDATA resource for backward compatibility.

IC_VERSION_CHECK

Used for backward compatibility to release version 1.0.

IC_VERSION_1_0

ICS Version 1.0.

IC_VERSION_1_2

ICS Version 1.2.

IC_VERSION_2_0

ICS Version 2.0.

IC_VERSION_2_02

ICS Version 2.02.
IC_VERSION_3_0
ICS Version 3.0.

Etc.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_RC_NODE data structure

ICSTD_ACTIVECHANNEL
Identifies the HIC_CHANNEL channel handle field of an active channel table.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_TABLE_FLAGS data type

ICSTD_ACTIVEPATH
Identifies the HIC_SESSION session (active path) handle field of an active path table.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_TABLE_FLAGS data type
ICSTD_ACTIVEPATHCHANNEL
Identifies the HIC_CHANNEL channel handle field of an active path table.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
IC_TABLE_FLAGS data type

ICSTD_CHANNEL
Identifies the channel ID field of a channel table.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
IC_TABLE_FLAGS data type

ICSTD_PATH
Identifies the path ID field of a path table.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
IC_TABLE_FLAGS data type

ICSTD_PATHCHANNEL
Identifies the channel ID field of a path table.
ICXVTCONFIG

This tag must be defined by XVT applications in order to use the ICS configuration API. Add the following line before \#include <xvt.h>.

\#define ICXVTCONFIG

○ WIN ○ XVT ○ DosLink

○ Accessory ○ Shell ○ Configurator

○ AIL ○ SL ○ EIL

ICXVTWIN

This tag must be defined by XVT applications that also include the WINDOWS.H include file. Add the following line before \#include <xvt.h>.

\#define ICXVTWIN

○ WIN ○ XVT ○ DosLink

○ Accessory ○ Shell ○ Configurator

○ AIL ○ SL ○ EIL

LPHIC_CHANNEL

Far pointer to an HIC_CHANNEL type.

○ WIN ○ XVT ○ DosLink

○ Accessory ○ Shell ○ Configurator

○ AIL ○ SL ○ EIL
ICS Data Structures/Types

**LPHIC_SESSION**
Far pointer to an **HIC_SESSION** type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**LPIC_RESULT_CONTEXT**
Far pointer to an **IC_RESULT_CONTEXT** type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

**LPIC_SINFO**
Far pointer to an **IC_SINFO** record type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:
**IC_SINFO** data structure

**LPIC_STATUSBUF**
Far pointer to an **IC_STATUSBUF** record type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL
See also:

IC_STATUSBUF data structure

LPIC_UPGRADE_INFO

Far pointer to an IC_UPGRADE_INFO record type.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_UPGRADE_INFO data structure

LPIC_VER_INFO

A far pointer to an IC_VER_INFO structure.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

IC_VER_INFO data structure
ICS Data Structures/Types

**NULL_HIC_CHANNEL**
ICS NULL channel handle.

- WIN
- XVT
- Accessory
- Shell
- AIL
- SL
- DosLink
- Configurator
- EIL

**NULL_HIC_CONFIG**
ICS NULL configuration handle.

- WIN
- XVT
- Accessory
- Shell
- AIL
- SL
- DosLink
- Configurator
- EIL

**NULL_HIC_SESSION**
ICS NULL session handle.

- WIN
- XVT
- Accessory
- Shell
- AIL
- SL
- DosLink
- Configurator
- EIL

**NULL_HIC_STATUSBUF**
ICS NULL extended status buffer handle.

- WIN
- XVT
- Accessory
- Shell
- AIL
- SL
- DosLink
- Configurator
- EIL

See also:

- IC_STATUSBUF data structure
ICS Data Structures/Types

NULL_IC_BUFHND
INFOConnect Connectivity Services NULL buffer handle type for shared data. Use this to test for the validity of an IC_BUFHND type.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

NULL_IC_MEMHND
INFOConnect Connectivity Services NULL buffer handle type for non-shared, intra-application memory. Use this to test for the validity of an IC_MEMHND type.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

PATHID
typedef struct {
  char ID[IC_MAXPATHIDSIZE];
} PATHID;

This data structure type defines a path ID.

ID
The path ID.

- WIN
- XVT
- DosLink

- Accessory
- Shell
- Configurator

- AIL
- SL
- EIL

VER_FILEDESCRIPTION_STR
Used in the Windows 3.1 version structure for all INFOConnect files that utilize the version definition. This value, which is a descriptive string of the component, must be defined by your component before including the icdef.rh file. See the Windows
3.1 Software Development Kit for information on updating the default version values in icdef.rh.

Note: The following is an example of an accessory resource for an application called MyApp.EXE that uses version control.

```c
#include "ver.h"
#define VER_FILETYPE VFT_APP
#define VER_FILESUBTYPE VFT2_UNKNOWN
#define VER_FILEDESCRIPTION_STR "MyApp Description"
#define VER_INTERNALNAME_STR "MyApp"

#include "Iicdef.rh"
/* Insert the rest of your RC file here */
```

WIN ● XVT ● DosLink

● Accessory ● Shell ● Configurator
● AIL ● SL ● EIL

See also:

VER_FILESUBTYPE data type
VER_FILETYPE data type
**VER_FILESUBTYPE**

Used in the Windows 3.1 version structure for all INFOConnect files that utilize the version definition. This value, which is a version subtype, must be defined by your component before including the `icdef.rh` file.

Valid subtype values are defined in `VER.H` from the Windows 3.1 SDK.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- **VER_FILEDESCRIPTION_STR** data type for an example

**VER_FILETYPE**

Used in the Windows 3.1 version structure for all INFOConnect files that utilize the version definition. This value, which is a version type, must be defined by your component before including the `icdef.rh` file.

Valid file-type values are defined in `VER.H` from the Windows 3.1 SDK.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- AIL
- SL
- EIL

See also:

- **VER_FILEDESCRIPTION_STR** data type for an example
VER_INTERNALNAME_STR

Used in the Windows 3.1 version structure for all INFOConnect files that utilize the version definition. This value, which is the internal name of the component, must be defined by your component before including the icdef.rh file.

- WIN
- XVT
- DosLink
- Accessory
- Shell
- Configurator
- All
- SL
- EIL

See also:

VER_FILEDESCRIPTION_STR data type for an example
Section 6
ICS Accessory Definition

An ICS accessory is an ICS application that can be invoked and controlled by other ICS applications. Accessories are written to be useful in building more sophisticated products. An accessory adheres to the following:

- Parse and react to the ICS-defined command line parameters.

Table 6–1. ICS Command Line Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;options file&gt;</td>
<td>The filename of the configuration options file. This file, when saved by the user, is to contain at least the most recent command line parameter options. This filename, if it exists, must be the first parameter on the command line.</td>
</tr>
<tr>
<td>-d or -D</td>
<td>Indicates running in debug mode, if the accessory supports it.</td>
</tr>
<tr>
<td>-k or -K</td>
<td>Followed by optional spaces and the current accessory’s ID. This option is always on the command line when the accessory is being invoked by an application (that is, the accessory is being executed as a result of a call to IcOpenAccessory or IcRunAccessory).</td>
</tr>
</tbody>
</table>
Table 6–1. ICS Command Line Parameters (cont.)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l or -L</td>
<td>Followed by optional spaces and the screen coordinates of the top left and bottom right window corners, enclosed in parentheses, as follows: (left,top,right,bottom)</td>
</tr>
<tr>
<td>-p or -P</td>
<td>Followed by optional spaces and the desired path name (path ID). When present, the accessory must open a session with the indicated path before any other processing. If an error is encountered (for example, an invalid options file format or an invalid command line parameter) the accessory must explicitly close the opened session.</td>
</tr>
<tr>
<td>-wxy or -Wxy</td>
<td>Window state parameter that is generated by a call to IcRunAccessory. Accessories must not define this command line parameter for its own purposes. Accessories must, however, recognize and react to the parameter. See the IcRunAccessory function in Section 6 for the valid values for x and y.</td>
</tr>
</tbody>
</table>

- Display the session name and the accessory's identification string on the accessory window's title bar.
- Register and deregister itself, obtaining a context number. See IcRegisterAccessory and IcDeregisterAccessory.
- Provide a .HIC include file that contains the accessory's context string and any accessory-specific statuses and errors.
- Respond to the IC_CONTROL_ACTIVATE status message by bringing itself into focus. If the accessory window was iconized, it should be restored.
- Delimit transactions using the IC_STATUS_TRANS status message.
- Terminate when all ICS sessions that it is using are closed.
- Provide the IC_RC_NODE type of user-defined resource in the resource file.
Adhere to the ICS on-line help style that follows:

- Follow the MS-Windows 3.1 Help guidelines. See the Windows 3.1 Software Development Kit for more information. In particular, the Help pull-down menu includes a Contents item, a Search for Help on item, a How to Use Help item, and an About item.

- Install the help book in the same directory as the executable. The help book should have the same root name as the executable with the HLP extension.

- Accessories that require ICS 2.0 or higher can use the Windows 3.1 help compiler, since the Windows 3.1 Help engine is redistributed with the Connectivity Services package.
Appendix A
Standard IDs (Keys) & Component Numbers

The following, case-insensitive IDs are standard for INFOConnect Connectivity Services. This means that any vendor developing an accessory or library that functions as described must associate itself with the corresponding ID through its resource file and install procedure. This ensures that accessories can access the desired runtime regardless of the vendor. This does not preclude the use of unique IDs where necessary.

Accessory IDs

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>VT-220 Emulator</td>
</tr>
<tr>
<td>DosLinkServer</td>
<td>DosLink Server Accessory</td>
</tr>
<tr>
<td>MT</td>
<td>A Series MT Emulator</td>
</tr>
<tr>
<td>PPT</td>
<td>Printer Pass Through for A Series</td>
</tr>
<tr>
<td>SNMP</td>
<td>SNMP Agent</td>
</tr>
<tr>
<td>UTS60</td>
<td>UTS60 Emulator</td>
</tr>
<tr>
<td>UTS60G</td>
<td>UTS60 Graphics Engine</td>
</tr>
</tbody>
</table>
### Service Library IDs

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMS</td>
<td>A Series COMS</td>
</tr>
<tr>
<td>DTPX</td>
<td>DTPX Service</td>
</tr>
<tr>
<td>HCLNTS</td>
<td>HLCN Terminal Services</td>
</tr>
<tr>
<td>INT1</td>
<td>2200 Interactive 1</td>
</tr>
<tr>
<td>TCP-A</td>
<td>A Series TCP/IP Access</td>
</tr>
<tr>
<td>TELCON</td>
<td>DCP TELCON</td>
</tr>
<tr>
<td>TELNET</td>
<td>TELNET Services</td>
</tr>
<tr>
<td>TP0</td>
<td>TCP TP0/RFC1006 Services</td>
</tr>
<tr>
<td>Trace</td>
<td>Trace INFOConnect Session Activity</td>
</tr>
<tr>
<td>TTY-1100</td>
<td>1100 Demand TTY</td>
</tr>
</tbody>
</table>


# External Interface Library IDs

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVGATE</td>
<td>A Series LAN Gateway</td>
</tr>
<tr>
<td>DosLink</td>
<td>ICS DosLink Access</td>
</tr>
<tr>
<td>Local</td>
<td>Local Communications</td>
</tr>
<tr>
<td>NetBIOS</td>
<td>NetBIOS Access</td>
</tr>
<tr>
<td>OSGATE</td>
<td>OS 2200 LAN Gateway</td>
</tr>
<tr>
<td>OSI</td>
<td>OSI Access</td>
</tr>
<tr>
<td>PS</td>
<td>Poll/Select Access</td>
</tr>
<tr>
<td>Stack</td>
<td>ICS Path Stacker</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP/IP Socket Access</td>
</tr>
<tr>
<td>TTY</td>
<td>TTY Access</td>
</tr>
<tr>
<td>UTS</td>
<td>Uniscope Access</td>
</tr>
<tr>
<td>WinSock</td>
<td>Windows Socket Access</td>
</tr>
<tr>
<td>XNS</td>
<td>XNS Access</td>
</tr>
<tr>
<td>X25</td>
<td>X.25 Access</td>
</tr>
</tbody>
</table>
Component Numbers

Component numbers are defined by the `IC_COMPONENT` data type and are used by the INFOConnect Connectivity Services configuration accessory to uniquely identify components.

Every INFOConnect component is assigned two `IC_COMPONENTs`: a *generic* `IC_COMPONENT` and a *branded* (supplier-specific) `IC_COMPONENT`. Components with non-zero generic `IC_COMPONENT`s conform to the interface defined by the specific component's .HIC include file. The branded `IC_COMPONENT` uniquely identifies the component. A component with a zero generic `IC_COMPONENT` performs some specific function defined by the vendor of that component.

The generic `IC_COMPONENT` identifies the component according to its function. The same generic `IC_COMPONENT` is used by all components that implement the same function. These types of components are preferable because they provide more flexibility and inter-operate with more current and future INFOConnect components. The developer provides the `IC_COMPONENT` value in the `IC_RC_NODE` of the component's resource file. If a non-zero generic `IC_COMPONENT` is not provided, the component is assumed to provide a unique function. No generic `IC_COMPONENT` value is assigned.

The branded `IC_COMPONENT` uniquely identifies the specific component. The vendor provides this value in the `IC_RC_NODE` of the component's resource file. If a non-zero branded `IC_COMPONENT` is not provided, a unique value will be generated and assigned when the library ID is added to the INFOConnect configuration database.

An `IC_COMPONENT` is constructed so that it can be managed using the universal Simple Network Management Protocol (SNMP) / Management Information Base (MIB). It consists of two parts: a component number and a supplier number. Generic `IC_COMPONENT`s and the supplier number of branded `IC_COMPONENT`s are assigned through the Malvern Development Group. Each vendor is responsible for managing the component number of the `IC_COMPONENT`s for its INFOConnect products. The currently assigned generic `IC_COMPONENT`s and branded supplier numbers are recorded in the `ic.hic` include file.
Notes:

- The Windows resource compiler does not accept LONG values in resources. The \texttt{IC\_COMPONENT} value must appear in the \texttt{IC\_RC\_NODE} resource as its two parts. Because of x86 little-endian architecture, the \texttt{IC\_COMPONENT} parts must be specified in the reverse order in the INFOConnect resource: supplier number followed by the component number.

- To obtain a vendor-specific, or branded, supplier number for your components, submit a Contact in the Primus database.
Appendix B
Status Types and Statuses

This section provides an overview of ICS standard statuses, which are also documented in Section 5, "ICS Data Structures and Types." The statuses are presented according to their use by components, as follows.

- Statuses Sent from Accessory to Library
- Statuses Sent from Library to Accessory
- Statuses Sent from Accessory to Accessory
- Statuses Sent from ICS to Accessory
- UTS-Specific Statuses
- DosLink-Specific Statuses
- Library Support for 1.11 Applications

Note that statuses may be described in multiple sections. Applications should process incoming events as needed. Service libraries and external interface libraries should produce the necessary status events when it is meaningful to do so.

Libraries may use these standard statuses or they may also generate their own, library-specific statuses using the library's context along with library-defined status types and status values. These values must be defined in the library's .HIC include file along with the context string to associate with the library's context. Applications wishing to recognize a library-specific status would include the particular library's .HIC include file. It would then be able to get the library's context from the context string by using \texttt{IcGetContext}. The application would recognize the library-specific status by retrieving the \texttt{IC\_RESULT\_CONTEXT} from the status.
Status Types and Statuses

Statues Sent from Accessory to Library

IC_STATUS_BLOCKING

IC_STATUS_BLOCKING is an application-initiated status type that is used to control the blocking mode of those service libraries that support it. The application toggles the blocking mode using the following statuses.

IC_BLOCKING_ON
Turn blocking on.

IC_BLOCKING_OFF
Turn blocking off.

Applications that require blocking should either be altered to support non-blocking interfaces or refuse to support a session over a library that sets sinfo.block_mode to FALSE.

IC_STATUS_BUFFER

IC_STATUS_BUFFER is an extended status that allows a buffer of information to be exchanged between an ICS application and an ICS library. It is to be used whenever the status information to be exchanged exceeds the bounds of the IC_RESULT structure. In this case, the IC_RESULT_VALUE portion of the status message is an INFOConnect buffer handle of type HIC_STATUSBUF. The IC_STATUSBUF data structure is defined as follows:

typedef struct aSTATUSBUF {
    IC_RESULT icstatus;
    IC_RESULT icerror;
    long reserved;
    unsigned uBufSize;
    unsigned uDataSize;
} IC_STATUSBUF;
Status Types and Statuses

`icstatus` The actual status message associated with the buffer of information.

`icerror` The `IC_RESULT` of the status request.

`reserved` Reserved for future use.

`uBufSize` The actual size, in bytes, of the data buffer.

`uDataSize` The size, in bytes, of the valid data in the data buffer.

Note that the data buffer must immediately follow the `IC_STATUSBUF` header. It should not contain pointers, but may contain offsets within the structure.

The `IC_STATUS_BUFFER` status message request can be synchronous or asynchronous. For a synchronous message, the library receives and processes the `IC_STATUS_BUFFER` status message, setting the `icerror` field to either `IC_OK` or an error. The `icerror` result is returned to the application as an `IC_StatusResult` message. An `IC_OK` result implies that the data buffer has been accessed and `uDataSize` is the size of the valid information.

For an asynchronous message, the library receives the `IC_STATUS_BUFFER` status message, sets the `icerror` field to `IC_INCOMPLETE` and returns `IC_INCOMPLETE`. The application receives the `IC_StatusResult` message with the `IC_INCOMPLETE` result. When the library finally supplies the `uDataSize` and accesses the data buffer, the `icerror` field should be set to `IC_COMPLETE` or to an error. The `IC_STATUS_BUFFER` status message should then be sent back to the application.

In both cases, the application should be responsible for freeing the data buffer. The data buffer should not be freed, however, until after the application receives an `IC_StatusResult` response of `IC_OK` or an error. If the result is `IC_INCOMPLETE`, this is the asynchronous case and the data buffer should not be freed until the status is returned via the `IC_STATUS_BUFFER` status message.

**IC_STATUS_CONNECT**

The `IC_STATUS_CONNECT` status type instructs the external interface library to alter the connection state. The connection states are defined by one of the following statuses.

4173 5390–000  B–3
Status Types and Statuses

IC_CONNECT_OPEN
Request to reopen the connection. This status is supported only if sinfo.reconnect is TRUE. If the library cannot honor this request, it should return an error from its IcSetStatus procedure.

IC_CONNECT_CLOSE
Close the connection. This status is supported only if sinfo.reconnect is TRUE.

IC_CONNECT_EOF
Request that no more data be sent (for use under TCP/IP). This status is supported only if sinfo.reconnect is TRUE.

IC_CONNECT_STATUS
Request that the EIL display status information to the user. This is the status sent when the user selects the Status button from the ICS Shell.

See the IC_STATUS_CONNECT entry in the Statuses Sent From Library to Accessory section below.

IC_STATUS_DATAFLAGS
This status controls the state of messages to mark expedited and/or multipart messages. The initial state of messages is assumed to be non-expedited and single part.

IC_DATAFLAGS(v)
A macro that creates an IC_DATAFLAGS status with value v.

IC_DATAFLAGS_EXPEDITED
This status indicates that the following transmitted message is urgent. It is to be delivered ahead of any other messages in the message queue.

IC_DATAFLAGS_MORE
This status indicates that the following transmitted messages are part of a multipart message.

IC_DATAFLAGS_NONE
This status indicates that none of the data flags are set. It is used to indicate the last part of a multipart message.
**IC_DATAFLAGS_RESERVED1**  
Reserved status.

**IC_DATAFLAGS_RESERVED2**  
Reserved status.

**IC_STATUS_FKEY**  
This application-initiated status type is used to send function key messages to the underlying layer of the ICS communication session. The function keys are defined by one of the following statuses.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_FKEY_BREAK</td>
<td>The break key</td>
</tr>
<tr>
<td>IC_FKEY_1</td>
<td>Function key 1</td>
</tr>
<tr>
<td>IC_FKEY_2</td>
<td>Function key 2</td>
</tr>
<tr>
<td>IC_FKEY_23</td>
<td>Function key 23</td>
</tr>
<tr>
<td>IC_FKEY_24</td>
<td>Function key 24</td>
</tr>
<tr>
<td>IC_FKEY_MSGWAIT</td>
<td>Uniscope-specific BEL character key</td>
</tr>
<tr>
<td>IC_FKEY_SYSMODE</td>
<td>Uniscope-specific OS3 system mode key used to set system mode</td>
</tr>
<tr>
<td>IC_FKEY_WSMODE</td>
<td>Uniscope-specific OS3 workstation mode key used to set workstation mode</td>
</tr>
</tbody>
</table>
Status Types and Statuses

IC_STATUS_REACTIVATE

These status messages must be sent by an application to the communication session by calling the set status procedure when sinfo.focus_notify is TRUE.

IC_REACTIVATE_ON  Application has received focus and sinfo.focus_notify is TRUE.

IC_REACTIVATE_OFF  Application has lost focus and sinfo.focus_notify is TRUE.

A library that needs to be notified of an application gaining/losing input focus (such as COMS), should set sinfo.focus_notify to TRUE in the IcLibGetSessionInfo procedure. (The COMS library generates and transmits messages (?on...) when the current window changes.)

Applications without visible windows must either support these statuses or refuse to support a session over a library that requests this type of notification.

IC_STATUS_TRANS

These status messages are sent by INFOConnect accessories to delimit transactions. Note that the initial state of all applications is assumed to be IC_TRANSACTION_OFF.

IC_TRANSACTION_ON  Indicates that IC_TRANSACTION_BEGIN and IC_TRANSACTION_END status will be sent.

IC_TRANSACTION_OFF  Indicates that IC_TRANSACTION_BEGIN and IC_TRANSACTION_END status will not be sent.

IC_TRANSACTION_BEGIN  Sent at the beginning of a transaction.

IC_TRANSACTION_END  Sent at the end of a transaction.
## Statuses Sent from Library to Accessory

**IC_STATUS_CONNECT**

These **IC_STATUS_CONNECT** statuses are typically issued from the EIL and report the state of the connection. The connection states are defined by one of the following statuses.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_CONNECT_OPEN</td>
<td>The logical connection is available for bidirectional communication under the current configuration.</td>
</tr>
<tr>
<td>IC_CONNECT_CLOSE</td>
<td>The logical connection is NOT available for bidirectional communication under the current configuration. This is the initial state of the session.</td>
</tr>
<tr>
<td>IC_CONNECT_EOF</td>
<td>The logical communication session is physically closed, no more data will be received (for use under TCP/IP).</td>
</tr>
<tr>
<td>IC_CONNECT_ACTIVITY</td>
<td>The physical connection (not necessarily this communication session) is functioning as expected.</td>
</tr>
<tr>
<td>IC_CONNECT_NOACTIVITY</td>
<td>The physical connection is NOT functioning as expected.</td>
</tr>
<tr>
<td>IC_CONNECT_BROKEN</td>
<td>Status that indicates that the other half of two connected sessions has closed. For example, a DosLink session receives this status when its partner session is closed.</td>
</tr>
<tr>
<td>IC_CONNECT_JOINED</td>
<td>Status that indicates that two sessions have been connected. For example, this is the status received when two DosLink sessions are connected.</td>
</tr>
</tbody>
</table>
### Status Types and Statuses

<table>
<thead>
<tr>
<th>Status Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IC_CONNECT_SERVER</strong></td>
<td>Status that originates from the server application (such as the DosLink Server accessory) that indicates readiness to the client. Libraries should send these statuses only when the status of the connection changes. See also the <strong>IC_STATUS_CONNECT</strong> entry in the &quot;Statuses Sent from Accessory to Library&quot; section.</td>
</tr>
</tbody>
</table>

### IC_STATUS_CONTROL

When initiated from an external interface library, a status of this type makes a request to the application. The requests are defined by one of the following statuses.

- **IC_CONTROL_ACTIVATE**<br>This status requests that the applications window become active for user input. This occurs when the user selects the GoTo button on the user interface window.

- **IC_CONTROL_RCVREADY**<br>This status requests that the application perform a receive request. It indicates to the application that a received message must be delivered or it may be lost.

- **IC_CONTROL_RCVAVAIL**<br>This is a notification, or advisory, status indicating that a message is available but not deliverable due to the state of the application. The session may be blocked until the message is delivered (for example, Poll/Select remains in the enqueued state until the message is delivered). See the **IC_STATUS_CONTROL** entry in the "Statuses Sent from Accessory to Accessory" section.

### IC_STATUS_LINESTATE

This EIL-initiated status type signifies the state of the underlying layer of the ICS communication session. This status is generally used by terminal emulators, such as MT and T27 type emulators. Therefore, Poll/Select libraries should generate these statuses.
An **IC_STATUS_LINESTATE** status message is generated by the external interface library each time the line state changes. Pass the event to the application by calling **IcMgrSendEvent(...)**. Applications receiving this event may or may not wish to process it.

The meaning of the line state statuses are as follows.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IC_LINESTATE_LCL</strong></td>
<td>The ICS communication session is neither transmitting nor receiving.</td>
</tr>
<tr>
<td><strong>IC_LINESTATE_RCV</strong></td>
<td>The ICS communication session is in receive mode.</td>
</tr>
<tr>
<td><strong>IC_LINESTATE_XMT</strong></td>
<td>The ICS communication session is in transmit or transmit/receive mode.</td>
</tr>
</tbody>
</table>
Statues Sent from Accessory to Accessory

IC_STATUS_DATAFLAGS

This application-initiated status controls the state of messages. It is used with the IcSetStatus/IcXmt functions and the IC_RcvDone event to mark expedited and/or multipart messages. The initial state of messages is assumed to be non-expedited and single part.

IC_DATAFLAGS(v)  A macro that creates an IC_DATAFLAGS status with value v.

IC_DATAFLAGS_EXPEDITED  This status indicates that the following transmitted message is urgent. It is to be delivered ahead of any other messages in the message queue.

IC_DATAFLAGS_MORE  This status indicates that the following transmitted messages are part of a multipart message.

IC_DATAFLAGS_NONE  This status is complementary to IC_DATAFLAGS_MORE. It is used to indicate the last part of a multipart message.

IC_DATAFLAGS_RESERVED1  Reserved status.

IC_DATAFLAGS_RESERVED2  Reserved status.

Note: For an example, see the IC_STATUS_DATAFLAGS status in Section 5, "Data Structures and Types".
IC_STATUS_CONTROL

This status makes a request to another connected accessory. The requests are defined by one of the following statuses.

**IC_CONTROL_ACTIVATE**

This status requests that the other application's window become active for user input.

**IC_CONTROL_RCVREADY**

This status requests that the other application perform a receive request.

See the IC_STATUS_CONTROL entry in the Statuses Sent From Library to Accessory section above.
Statuses Sent from ICS to Accessory

**IC_STATUS_COMMGR**

This status type originates from the ICS Manager itself and conveys initialization or termination information. If the ICS Manager terminates, the ICS accessory must call the ICS initialization routine before calling any other ICS procedures.

**IC_COMMGR_INITIALIZED**

Status sent to all Windows applications when the ICS Manager finishes initializing. ICS accessories may now call the ICS initialization routine, if necessary, before establishing INFOConnect sessions.

**IC_COMMGR_TERMINATED**

Status sent to all Windows applications when the ICS Manager finishes terminating. All ICS accessories should either close or call the ICS initialization routine before establishing another ICS session.

**IC_COMMGR_QUERYEXIT**

Status sent to all ICS communications sessions when the user closes the INFOConnect Shell. If the application does not wish to close the session, it should cancel the exit by calling `IcExitOk(FALSE)`. Otherwise, call `IcExitOk(TRUE)`.

**IC_COMMGR_QUERYSHUTDOWN**

Status sent to all ICS communications sessions when Windows is exiting. If the application does not wish to close the session, it should cancel the exit by calling `IcExitOk(FALSE)`. Otherwise, call `IcExitOk(TRUE)`.
### Status Types and Statuses

<table>
<thead>
<tr>
<th>Status Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IC_COMMMGR_CANCELEXIT</strong></td>
<td>Status sent to all ICS communications sessions that previously received an <strong>IC_COMMMGR_QUERYEXIT</strong> status when at least one of the applications called IcExitOk(FALSE).</td>
</tr>
<tr>
<td><strong>IC_COMMMGR_EXIT</strong></td>
<td>Status sent to all ICS communications sessions if IcExitOk(FALSE) is never called. The ICS Manager will then exit.</td>
</tr>
<tr>
<td><strong>IC_COMMMGR_REINSTALL</strong></td>
<td>Status posted to all windows by install.exe when the ICS Manager is being reinstalled.</td>
</tr>
</tbody>
</table>
UTS-Specific Statuses

IC_STATUS_UTS
- IC_UTS_SELECTION
- IC_UTS_DVC_READY
- IC_UTS_DVC_BUSY
- IC_UTS_DVC_ERROR
- IC_UTS_DVC_NOTREADY
- IC_UTS_ATTENTION
- IC_UTS_DESELECT_ACTIVITY
- IC_UTS_DESELECT_DID
- IC_UTS_MSGWAIT
- IC_UTS_POC

This status type may be used to send and receive special messages to/from the UTS external interface library (and the INT1 SL).

The library may send the following status to the application. The IC_RESULT_VALUE is interpreted as two subfields: IC_RESULT_SUBTYPE (subtype) and IC_RESULT_SUBVALUE (subvalue). A special macro, IC_MAKE_UTS_RESULT(t, v), is available to create an IC_RESULT from the standard context and from the IC_RESULT_TYPE and IC_RESULT_VALUE.

IC_UTS_SELECTION subtype 0
- IC_UTS_DESELECT_ACTIVITY This status message has a subvalue of 0x71. It is a request to deselect the current device.
- IC_UTS_DESELECT_DID This status message has a subvalue of 0x72. It is a request to flush and deselect the current device.
- IC_UTS_MSGWAIT This status message has a subvalue of 0x07. This is message wait.

Subvalues in the range of 0x20 - 0x6F and 0x73 - 0x7F These status messages request the selection of the given Device ID (DID).

The application may send the following status to the UTS external interface. The IC_RESULT_VALUE is interpreted as two subfields: IC_RESULT_SUBTYPE (subtype) and IC_RESULT_SUBVALUE (subvalue).
Status Types and Statuses

**IC_UTS_DVC_READY subtype 0x10**
Subvalues in the range of
0x20 - 0x6F and 0x73 - 0x7F
These status messages indicate that
the given device (DID) is ready.

**IC_UTS_DVC_BUSY subtype 0x11**
Subvalues in the range of
0x20 - 0x6F and 0x73 - 0x7F
These status messages indicate that
the given device (DID) is busy.

**IC_UTS_DVC_ERROR subtype 0x12**
Subvalues in the range of
0x20 - 0x6F and 0x73 - 0x7F
These status messages indicate that
the given device (DID) has an error.

**IC_UTS_DVC_NOTREADY subtype 0x13**
Subvalues in the range of
0x20 - 0x6F and 0x73 - 0x7F
These status messages indicate that
the given device (DID) is not
responding.

**IC_UTS_ATTENTION subtype 0x20**
**IC_UTS_POC**
This status message has a subvalue of
0x36. It indicates power confidence
tests have completed (that is, send
<DLE>6 to the host).
DosLink-Specific Statuses

DOSLINK_SINFO
This status type, when associated with the DosLink EIL context, is sent from the DosLink Server accessory by calling `IcMgrSetResult`. The value of the status is the session handle on which to retrieve SINFO data. The DosLink EIL uses the result value as the session handle for calling `IcMgrGetSessionInfo`. The SINFO record is then passed to the DosLink Client using the DosLink `IcSetServerInfo` API. When the SINFO data has been copied to the DosLink Client session, an IC_CONNECT_SERVER (IC_STATUS_CONNECT type) status is sent to the client session. The session information data is then available to the client session.
Library Support for 1.11 Applications

Applications written with the 1.11 version of the IDK use the `IC_STATUS_SPECIALMSG` status message instead of the `IC_STATUS_UTS` or the `IC_STATUS_FKEY` statuses. In order for 2.0 libraries to support these applications, they should be aware of this.

The `IC_STATUS_SPECIALMSG` status with `IC_RESULT_VALUE 0x07` (Message Wait), has the same binary value as the new `IC_UTS_MSGWAIT` status. The `IC_UTS_DESELECT...` statuses also have the same binary values as their `IC_STATUS_SPECIALMSG` counterparts. Therefore, libraries need not do any special processing for sending these status to version 1.11 applications.

Version 1.11 applications will be sending `IC_STATUS_SPECIALMSG` type statuses to the library. If the library receives an `IC_UTS_SELECTION` (subtype == 0) status message from an application, the library should use the `IC_RESULT_SUBVALUE` to perform the `IC_STATUS_FKEY` action using the following table.

<table>
<thead>
<tr>
<th>SUBVALUE</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x37</td>
<td>IC_FKEY_1</td>
</tr>
<tr>
<td>0x47</td>
<td>IC_FKEY_2</td>
</tr>
<tr>
<td>0x57</td>
<td>IC_FKEY_3</td>
</tr>
<tr>
<td>0x67</td>
<td>IC_FKEY_4</td>
</tr>
<tr>
<td>0x20 to 0x32</td>
<td>IC_FKEY_5 to IC_FKEY_22</td>
</tr>
</tbody>
</table>

**UTS EIL (and INT1 SL)**

<table>
<thead>
<tr>
<th>SUBVALUE</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07</td>
<td>IC_FKEY_MSGWAIT</td>
</tr>
</tbody>
</table>

**TTY EIL**

<table>
<thead>
<tr>
<th>SUBVALUE</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IC_FKEY_BREAK</td>
</tr>
</tbody>
</table>
Status Types and Statuses

The \texttt{IC\_STATUS\_SPECIALMSG} status is presented below for completeness. Existing applications should be modified to use the \texttt{IC\_STATUS\_UTS} and \texttt{IC\_STATUS\_FKEY} statuses before release 3.0 of the IDK.

\textbf{IC\_STATUS\_SPECIALMSG}

This status type was used by the 1.11 version of some of the ICS layers to send and receive special messages through the communication session. The unique status values are defined as follows.

\textbf{TTY EIL}

TTY external interface library interprets the following \texttt{IC\_STATUS\_SPECIALMSG IC\_RESULT\_VALUE}, sent by an application, as follows.

\begin{itemize}
\item 0x00 Break key.
\end{itemize}

The application uses the \texttt{IC\_MAKE\_RESULT} macro with \texttt{IC\_RESULT\_CONTEXT\_STD, IC\_STATUS\_SPECIALMSG}, and value 0x00 to create this status before calling the set status procedure.

\textbf{From UTS EIL or INT1 SL to the Accessory}

The UTS external interface library and the INT1 service library generate the following \texttt{IC\_STATUS\_SPECIALMSG IC\_RESULT\_VALUE}s to an application.

\begin{itemize}
\item 0x07 Unsolicited MESSAGE WAIT from host.
\item 0x72 Deselection DID has been received from host.
\end{itemize}

The application can use the \texttt{IC\_GET\_RESULT\_TYPE} and \texttt{IC\_GET\_RESULT\_VALUE} macros to examine the status result.

\textbf{From Accessory to UTS EIL or INT1 SL}

The UTS external interface library and the INT1 service library interpret the following \texttt{IC\_STATUS\_SPECIALMSG IC\_RESULT\_VALUE}s from an application as follows.

\begin{itemize}
\item 0x07 Message Wait.
\item 0x37 F1 Key.
\end{itemize}
<table>
<thead>
<tr>
<th>Status Code</th>
<th>Key Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x47</td>
<td>F2 Key.</td>
</tr>
<tr>
<td>0x57</td>
<td>F3 Key.</td>
</tr>
<tr>
<td>0x67</td>
<td>F4 Key.</td>
</tr>
<tr>
<td>0x20</td>
<td>F5 Key.</td>
</tr>
<tr>
<td>0x21</td>
<td>F6 Key.</td>
</tr>
<tr>
<td>0x22</td>
<td>F7 Key.</td>
</tr>
<tr>
<td>0x23</td>
<td>F8 Key.</td>
</tr>
<tr>
<td>0x24</td>
<td>F9 Key.</td>
</tr>
<tr>
<td>0x25</td>
<td>F10 Key.</td>
</tr>
<tr>
<td>0x26</td>
<td>F11 Key.</td>
</tr>
<tr>
<td>0x27</td>
<td>F12 Key.</td>
</tr>
<tr>
<td>0x28</td>
<td>F13 Key.</td>
</tr>
<tr>
<td>0x29</td>
<td>F14 Key.</td>
</tr>
<tr>
<td>0x2A</td>
<td>F15 Key.</td>
</tr>
<tr>
<td>0x2B</td>
<td>F16 Key.</td>
</tr>
<tr>
<td>0x2C</td>
<td>F17 Key.</td>
</tr>
<tr>
<td>0x2D</td>
<td>F18 Key.</td>
</tr>
<tr>
<td>0x2E</td>
<td>F19 Key.</td>
</tr>
<tr>
<td>0x2F</td>
<td>F20 Key.</td>
</tr>
<tr>
<td>0x30</td>
<td>F21 Key.</td>
</tr>
<tr>
<td>0x31</td>
<td>F22 Key.</td>
</tr>
</tbody>
</table>

The application uses the **IC_MAKE_RESULT** macro with **IC_RESULT_CONTEXT_STD, IC_STATUS_SPECIALMSG**, and the desired value from above to create the status result before calling the set status procedure.
Appendix C
Errors and Results

This appendix lists and describes the INFOConnect Connectivity Services errors and informative results, as well as standard configuration accessory errors and errors specific to Unisys-provided ICS service libraries and external interface libraries. These fields must be provided by other vendors developing the given library.

Library-specific errors are generated using the library's context along with library-defined error types and error values. These values are defined in the library's .HIC include file along with the context string associated with the library's context. (The context string must be unique up to the first eight characters.) To maintain flexibility, applications should generally not be coded to particular library-specific errors. However, those developer's wishing to recognize a library-specific error would include that particular library's .HIC include file into the application. The application would then be coded to retrieve the library's context from the context string using IcGetContext. The library-specific error is recognized using the IC_GET_RESULT... API to retrieve various parts from the error result, including the IC_RESULT_CONTEXT.
INFOConnect Connectivity Services

ICS Standard Errors

The following error results are common/general errors defined for INFOConnect Connectivity Services. They may be returned as the result of a procedure call or with an error event (IC_Error, IC_RcvError, and so forth under MS-Windows or E_IC_ERROR, and so forth under XVT).

Most errors (with the exception of interactive library configuration and the Version 2.0 implementation of the EIL AutoDial feature in TTY, for example) are filtered back through the application. The application may be coded to handle the error itself, perhaps by displaying it to the user or by performing some other action, or the error may be passed back to INFOConnect by calling the INFOConnect default error procedure.

Library developers may use any of the standard error results, but must call IcSetSessionError prior to exiting the active procedure. (See IcSetSessionError in Section 3, "INFOConnect API", for more information.)

Terminate-type errors indicate that the particular request failed and that all other requests on the associated session will also fail. Therefore, the communication session must be closed. If the default error procedure is called, the error message will be displayed to the user and the communication session will be closed automatically.

Severe-type errors indicate that a particular request failed. Errors in the range of this type are serious enough that they are always displayed to the user.

Errors in the range of the IC_ERROR_WARNING type indicate that the request succeeded and suggest that the result should either be displayed to the user or logged by the application for future reference. User intervention (for example, reconfiguring or upgrading the software) will prevent the warning from reoccurring.

Errors in the range of the IC_ERROR_INFO type are informative. They may be optionally logged by the application and should not be displayed to the user. Results that do not indicate an error, but rather some return condition, are also of type IC_ERROR_INFO.

IC_ASSIGNMENT_ERROR  (Value 902)

  The template ID is already assigned to a template.
  The requested update has not been made.

Level: Severe
**IC_ASSIGNMENT_ERROR** indicates that the requested template ID cannot be assigned. This error occurs when an attempt is made to use a template ID that is already assigned to a template.

The user seeing this error can either use a different template ID or can rename the existing template before retrying the action.

**IC_ASSIGNMENT_UPDATED (Value 2004)**

The template ID has been updated.

Level: **Informational**

**IC_ASSIGNMENT_UPDATED** result indicates that the request to update to a template ID that has been previously assigned to a template has been completed.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

**IC_CANCELED (Value 2003)**

The user cancelled from the dialog.

Level: **Informational**

The **IC_CANCELED** result indicates that the user cancelled from the active dialog.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.
Errors and Results

**IC_COMPLETE** (Value 2013)

The pending request has successfully completed.

Level: **Informational**

The **IC_COMPLETE** result indicates that a pending request has been completed. This is the result used to identify the completion of an extended, asynchronous status request. See Appendix B for information on extended statuses.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

**IC_CONTEXT_ALREADY_CREATED** (Value 701)

The ICS context for `<context string>` has already been created.

Context strings must be unique.

Level: **Severe**

**IC_CONTEXT_ALREADY_CREATED** indicates that the requested INFOConnect context already exists. This error occurs when an attempt is made to load the INFOConnect component with the given context string and a component with that same context string has already been loaded. Each context string must be unique.

The user seeing this error should unload the existing component before trying to load the component with the same context string.

**IC_CONTEXT_ALREADY_DELETED** (Value 702)

The ICS context for `<context>` has already been deleted.

Contact the component’s vendor for further information.

Level: **Severe**

**IC_CONTEXT_ALREADY_DELETED** indicates that a request to delete a context cannot be completed. This error occurs when the given context does not exist because either it was already deleted or it was never created. This error should not occur in the released version of a product.

The user seeing this error should contact the component’s vendor for further information.

**IC_CONTEXT_INVALID** (Value 703)

Invalid Context: `<context>`.

Contact the component’s vendor for further information.
Level: **Severe**

**IC_CONTEXT_INVALID** indicates an invalid context has been detected. This error occurs when an attempt is made to access the given context and that context has not been successfully initialized. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor for further information.

**IC_CONTEXT_NOT_FOUND** (Value 704)

Context `<context>` not found.
Contact the component's vendor for further information.

Level: **Severe**

**IC_CONTEXT_NOT_FOUND** indicates that the given context cannot be found in the INFOConnect table of contexts. This error occurs when an attempt is made to access a component whose context was not successfully created. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor for further information.

**IC_CONTEXTSTRING_NOT_FOUND** (Value 705)

Context string `<context string>` not found.
Contact the component's vendor for further information.

Level: **Severe**

**IC_CONTEXTSTRING_NOT_FOUND** indicates that the given context string cannot be found in the INFOConnect table of contexts. This error occurs when an attempt is made to access a component whose context was not successfully created. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor for further information.

**IC_CONTEXTSTRING_TRUNCATED** (Value 706)

Context string `<context string>` truncated.
Contact the component's vendor for further information.

Level: **Severe**
Errors and Results

**IC_CONTEXTSTRING_TRUNCATED** indicates that the retrieved context string was truncated. This error occurs when an attempt is made to retrieve the context string from a context and the output buffer parameter is not big enough to hold the context string. The buffer should be at least 9 bytes big.

The user seeing this error should contact the component's vendor for further information.

**IC_CONTEXTTABLE_FULL** (Value 700)

The context table is full. Close some Windows applications and retry the action.

Level: Severe

**IC_CONTEXTTABLE_FULL** indicates that no more context entries can be added to the table of context/context strings. This error occurs in low memory conditions.

The user seeing this error should close some Windows applications and try the action again.

**IC_ERROR_ACCESSORY_FAILED** (Value 801)

Accessory <name> execution failed. Verify that this is a valid Windows code file.

Level: Severe

**IC_ERROR_ACCESSORY_FAILED** indicates that the given accessory could not be executed. This error occurs when an attempt is made to execute an INFOConnect application as an accessory and the accessory cannot execute successfully (see the IcOpenAccessory and IcRunAccessory functions).

The user seeing this error should verify that the given file is a valid Windows code file.

**IC_ERROR_ACCESSORY_NOT_FOUND** (Value 800)

Accessory <name> not found. Verify the accessory installation, the file name, and the DOS path.

Level: Severe

**IC_ERROR_ACCESSORY_NOT_FOUND** indicates that the given file could not be located. This error occurs when an attempt is made to execute an INFOConnect application as an accessory and the accessory cannot execute successfully (see the IcOpenAccessory and IcRunAccessory functions).
accessory through `IcOpenAccessory` or `IcRunAccessory` and the file or the DOS path could not be found.

The user seeing this error should verify that the given file name and DOS path, as well as the ICS installation of the accessory.

**IC_ERROR_ALREADYCLOSED** (Value 509)

The ICS Communications Manager has already been terminated.
Close and restart the INFOConnect Shell.

Level: **Severe**

`IC_ERROR_ALREADYCLOSED` indicates that no INFOConnect Shell is executing. This error occurs only from `IcTerminateShell` when `IcInitShell` has not been called. ICS Shell developers must be sure to call `IcInitShell` and `IcTerminateShell` in pairs. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

**IC_ERROR_APP_BUSY** (Value 11)

Application queue full. Message discarded.

Level: **Severe**

`IC_ERROR_APP_BUSY` indicates that a message cannot be posted to an application. This error occurs when the application's message queue is full.

The user seeing this error should give control to the application so that some messages may be delivered.

**IC_ERROR_APP_GONE** (Value 12)

Application queue closed. Message discarded.

Level: **Severe**

`IC_ERROR_APP_GONE` indicates that a message cannot be posted to an application. This error occurs when an AIL/IIL attempts to post a message to an application whose window handle is no longer valid. This may occur if the application terminates without closing all of its INFOConnect sessions.

The user seeing this error should contact the application's vendor.
Errors and Results

**IC_ERROR_BADFUNCTION (Value 300)**

Internal error: Bad function. Contact the component's vendor for further information.

Level: **Severe**

For ICS DosLink applications, **IC_ERROR_BADFUNCTION** indicates an internal error to the DosLink.386 driver.

The user seeing this error should contact the component's vendor.

**IC_ERROR_BADPARAMETER (Value 4)**

Invalid parameter received. Contact the component's vendor for further information.

Level: **Severe**

**IC_ERROR_BADPARAMETER** is returned when an ICS procedure receives an invalid parameter. This may occur when an unexpected NULL string pointer is received or when a buffer length is less than the minimum required by the called procedure. Errors of this type should not occur in the final release of a product.

The user seeing this error should contact the component's vendor.
Errors and Results

IC_ERROR_BADREVISION (Value 302)

This component references unknown revision <number>. Reboot the computer and try again.

Level: Severe

IC_ERROR_BADREVISION indicates that the given revision number is unknown to the ICS Manager. It may occur when a component contains an invalid or unknown revision number or as the result of memory corruption.

The user seeing this error should attempt to recreate it before contacting the component's vendor.

IC_ERROR_BADSESSION (Value 1)

Invalid session handle detected at <string>. Session must be terminated.

Level: Termination

IC_ERROR_BADSESSION indicates that an invalid session handle has been detected. This error occurs when some underlying layer of INFOConnect Connectivity Services receives a handle to a session that is not a valid session handle or, where required, the handle of an established session. Once the error message is displayed, the communication session is to be closed through the close session procedure. If the default error procedure is called, it closes the session automatically.

For debugging purposes, library developers returning this error result from an ICS library must first call IcSetSessionError with the lpinsert1 parameter pointing to a string that identifies the location in the code where the error was detected (for example, IcLibXmt).

The user seeing this error should turn on the Tracing Log facility from the INFOConnect manager for the session and recreate the error. The resulting log file should be sent to the INFOConnect support representative.
Errors and Results

IC_ERROR_BADTEMPLATE (Value 611)

Configuration of path template <template name> is invalid. Choose Modify from Install Path Templates to update the corrupted data.

Level: Severe

IC_ERROR_BADTEMPLATE is an internal error indicating that the specified path template configuration is corrupted. It may occur as the result of disk corruption.

The user seeing this error should modify the template and save it, allowing the corrupted data to be overwritten.

IC_ERROR_BADVERSION (Value 301)

This component references unknown version <number>. Reboot the computer and try again.

Level: Severe

IC_ERROR_BADVERSION indicates that the given version number is unknown to the ICS Manager. It may occur when a component contains an invalid or unknown version number or as the result of memory corruption.

The user seeing this error should attempt to recreate it before contacting the component's vendor.

IC_ERROR_CANCELOPEN (Value 2000)

User did not select a valid path identification.

Level: Informational

The IC_ERROR_CANCELOPEN result occurs when the user selects the Cancel button on the Select Path dialog box during the open session procedure. Note that this is an informational result that indicates the dialog box was successfully executed and that a path was NOT selected by the user.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

IC_ERROR_CHAN_BUSY (Value 612)

Library <name> is still busy opening channel <identifier>. Wait until the channel has opened and try again.

Level: Severe
IC_ERROR_CHAN_BUSY indicates that a library’s IcLibOpenSession routine is being called before a previous call to the library’s IcLibOpenChannel routine for that same channel has been completed. This may occur, for instance, if IcLibOpenChannel waits for user input.

The user seeing this error should wait until the channel finishes opening and try to open the path again.

IC_ERROR_CHANNELINUSE (Value 503)
Channel <identifier> already in use by <library> non-multiplexing library.
The requested path cannot be opened at this time.

Level: Severe

IC_ERROR_CHANNELINUSE indicates that the given ICS path (specified by <identifier>) cannot be opened a second time. This error occurs from IcOpenSession when the channel configured for the given ICS path is already being used by the non-multiplexing library specified by <library>.

The user seeing this error should close one of the applications.

IC_ERROR_COLON_PRESENT (Value 906)
Colon(:) not allowed in ID. Correct the ID and try again.

Level: Severe

IC_ERROR_COLON_PRESENT indicates that the ICS ID erroneously contains a colon. ICS does not allow colons in IDs.

The user seeing this error should correct the ID and try the action again.
IC_ERROR_INITICS (Value 500)

Unable to start INFOConnect. ABORTING.
Contact the INFOConnect support representative for further information.

Level: Severe

IC_ERROR_INITICS indicates that INFOConnect cannot be executed. This error occurs from IcInitIcs when an unknown error occurs during INFOConnect Connectivity Services initialization. INFOConnect is aborted.

The user seeing this error should contact the INFOConnect support representative.

IC_ERROR_INMODIFY (Value 507)

Path <name> is currently being modified.
You cannot establish a session with this path.

Level: Severe

IC_ERROR_INMODIFY indicates that the given path cannot be opened. This error occurs when the user is modifying a path and, at the same time, attempts to open a session over it. These two activities are mutually exclusive. The session will not be established.

The user seeing this error should finish modifying the path configuration before attempting to use it.

IC_ERROR_INTERNAL (Value 5)

Internal error detected at <string>.
Contact the component vendor for further instruction.

Level: Severe

IC_ERROR_INTERNAL indicates a non-fatal internal error has been detected. This error occurs when some layer of ICS detects an impossible or unlikely state. For debugging purposes, developers returning this error from an ICS library must first call IcSetSessionError with the lpinsert1 parameter pointing to a location identification string.

The user seeing this error should contact the component's vendor for more information.
IC_ERROR_INVALID_CONFIGREC (Value 900)

Invalid configuration record structure returned. Configuration aborted. Select Configure from the Configure Packages window.

Level: Severe

IC_ERROR_INVALID_CONFIGREC indicates that a configuration record was invalid. This error occurs when the structure of the record does not match that expected by the ICS database.

The user seeing this error should select Configure from the Configure Packages window to force a data upgrade to occur. The action should then be retried. If the error still occurs, contact the component's vendor.

IC_ERROR_INVALIDPATH (Value 502)

Invalid path requested: <path ID>. Verify the path configuration.

Level: Severe

IC_ERROR_INVALIDPATH indicates that the given path ID is invalid. This error occurs when the user tries to establish a session with a path ID that is not configured.

The user seeing this error should verify that a path with the given path ID is properly configured.

IC_ERROR_INVALID_WINCOMBO (Value 8)

Invalid window state combination. Contact the component's vendor for further information.

Level: Severe

IC_ERROR_INVALID_WINCOMBO indicates that a request was made to open an ICS accessory with a hidden/active or maximized/background window state. This error occurs when one of these invalid combinations of window state options is passed to IcOpenAccessory or IcRunAccessory (through the -W option). See IcOpenAccessory or IcRunAccessory in Section 3, "INFOConnect API", for more information. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.
Errors and Results

IC_ERROR_INVALID_WINOPTION (Value 7)

Invalid window state option. Contact the component's vendor for further information.

Level: Severe

IC_ERROR_INVALID_WINOPTION indicates that a request was made to open an ICS accessory using unknown window state options. This error occurs when an invalid window state option is passed to IcOpenAccessory or IcRunAccessory (through the -W option). See IcOpenAccessory or IcRunAccessory in Section 3 of the ICS Reference Manual, "INFOConnect API", for more information. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_ERROR_LIBRARY_CONFIG (Value 901)

The given ID is still configured in a path. Delete this path or reconfigure it without this library.

Level: Severe

IC_ERROR_LIBRARY_CONFIG indicates that the library cannot be deleted. This error occurs when an attempt is made to delete a library while it is still configured in a path.

The user seeing this error should delete or reconfigure the path without the offending library ID before deleting the library.

IC_ERROR_LOSTRCV (Value 305)

Receive request lost. Please retry the request.

Level: Severe

For ICS DosLink applications, IC_ERROR_LOSTRCV indicates that a receive request was lost.

The user seeing this error should retry the receive request.
IC_ERROR_LOSTXMT (Value 306)

Transmit request lost. Please retry the request.

Level: Severe

For ICS DosLink applications, IC_ERROR_LOSTXMT indicates that a transmit request was lost.

The user seeing this error should retry the transmit request.

IC_ERROR_MGR_BUSY (Value 9)

Communication queue full. Request ignored.

Level: Severe

IC_ERROR_MGR_BUSY indicates that a message cannot be posted. This error occurs when the ICS Communications Manager message queue is full. The accessory must relinquish control so that some messages may be delivered.

The user seeing this error should close the accessory to allow messages to be delivered and contact the accessory's vendor.

IC_ERROR_NEWREVISION (Value 615)

This component requires a newer version (<number>) of the INFOConnect Communications Manager. Update the ICS software before using this component.

Level: Severe

IC_ERROR_NEWREVISION indicates that the component cannot execute with the installed ICS Manager. This error occurs when a newer revision of an ICS accessory or application attempts to run with an older version of ICS.

The user seeing this error must update the ICS software before using the component.
Errors and Results

IC_ERROR_NEWVERSION (Value 605)

This application requires Version <number> of the INFOConnect Communications Manager. Update the ICS software before using this component.

Level: Severe

IC_ERROR_NEWVERSION indicates that the component cannot execute with the installed ICS Manager. This error occurs when a newer version of an ICS accessory or application attempts to run with an older version of ICS.

The user seeing this error must update the ICS software before using the calling application.

IC_ERROR_NOCHANDATA (Value 609)

Channel <identifier> configuration data for library <library> missing.

Have the Administrator modify the channel configuration for this library.

Level: Severe

IC_ERROR_NOCHANDATA is an internal error indicating that the specified library is missing the specified channel configuration data. This may occur as the result of disk corruption.

The user seeing this error should modify the channel configuration.

IC_ERROR_NOCLOSE (Value 508)

The ICS Communications Manager is not ready to terminate. Be sure that all dialogs are closed.

Level: Severe

IC_ERROR_NOCLOSE indicates that the ICS Manager cannot be closed. This error occurs, for example, when the user still has the Select Path dialog open.

The user seeing this error should close all ICS dialogs before closing INFOConnect.

IC_ERROR_NODATABASE (Value 102)

Database Not Found. Please terminate and restart INFOConnect.

Level: Termination

IC_ERROR_NODATABASE indicates that a valid INFOConnect database could not be located. This error occurs when the database was not properly opened or
created. During initialization, the ICS Shell would have received the specific database error and should have displayed the error to the user. Once the error message is displayed, INFOConnect should be terminated and restarted.

The user seeing this error should verify that all INFOConnect command line parameters are correct. If the problem still occurs, contact the ICS Shell vendor.

**IC_ERROR_NOFIND** (Value 2008)

The requested information could not be found.

**Level:** Informational

The **IC_ERROR_NOFIND** result indicates that requested information could not be found.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

**IC_ERROR_NOLIBLOAD** (Value 600)

Unable to load <component>. The result code is <number>. Verify that this is a valid Windows code file.

**Level:** Severe

**IC_ERROR_NOLIBLOAD** indicates that the ICS component specified by <component> cannot be loaded. A result code is specified by <number>.

The user seeing this error should verify that the given component is a valid Windows code file and that Windows itself is properly installed.

**IC_ERROR_NOLIBRARY** (Value 607)

Library <name> is not installed. Install the necessary library and try again.

**Level:** Severe

**IC_ERROR_NOLIBRARY** indicates that the specified library is not currently installed. This may occur if the Trace library is deinstalled or deleted and the user attempts to trace sessions. The error may also occur if the Local library is deinstalled or deleted and an application attempts to use an accessory.

The user seeing this error should install the necessary library.
Errors and Results

IC_ERROR_NOMEMORY (Value 3)

Memory Error. Free some memory and try again.

Level: Severe

IC_ERROR_NOMEMORY is returned when an attempt to allocate or access an ICS memory block fails. It occurs in low memory conditions.

The user seeing this error should free some memory before continuing.

IC_ERROR_NOPARTNER (Value 303)

Partner session could not be found.

Level: Warning

IC_ERROR_NOPARTNER indicates that the partner session (for example, for an ICS DosLink Client/Server application) is not yet established.

The user seeing this error should wait until the partner session establishes before continuing to use the session.
IC_ERROR_NOPATHDATA (Value 608)

Path configuration data for library <name> is missing. Modify the path configuration and try again.

Level: Severe

IC_ERROR_NOPATHDATA is an internal error indicating that the specified library is missing path configuration data. It may occur as the result of disk corruption.

The user seeing this error should modify the path configuration.

IC_ERROR_NOPATHID (Value 903)

Path ID missing. Verify the path ID and try again.

Level: Severe

IC_ERROR_NOPATHID indicates that the path ID is not found. It occurs when an attempt is made to access a path with an ID that is not assigned.

The user seeing this error should verify that the path ID is correct and retry the action.

IC_ERROR_NORCVMEM (Value 309)

Internal error: no receive memory. Free some memory and try again.

Level: Severe

For ICS DosLink applications, the IC_ERROR_NORCVMEM internal error is returned when an attempt to allocate or access an ICS memory block fails. It occurs in low memory conditions.

The user seeing this error should free some memory before continuing.
Errors and Results

IC_ERROR_NOSESSION (Value 2001)

    Session is not established.

    Level: Informational

IC_ERROR_NOSESSION result indicates that the session in question has not yet been successfully established. The session is in the process of opening, and may or may not open successfully.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

IC_ERROR_NOSESSIONMEM (Value 307)

    Internal error: no session memory. Free some memory and try again.

    Level: Severe

For ICS DosLink applications, IC_ERROR_NOSESSIONMEM is returned when an attempt to allocate or access an ICS memory block fails. It occurs in low memory conditions.

The user seeing this error should free some memory before continuing.

IC_ERROR_NOTEMPLATE (Value 610)

    Path template <template ID> is not configured. Try to reconfigure the template.

    Level: Severe

IC_ERROR_NOTEMPLATE is an internal error indicating that a path is attempting to use the specified path template that does not exist. It may occur as the result of disk corruption.

The user seeing this error should try to reconfigure the given template. If the error still occurs, contact the INFOConnect Configuration Accessory vendor.
IC_ERROR_NOVERSION (Value 603)

Cannot verify version information. <name> not loaded. Try reinstalling this product.

Level: Severe

IC_ERROR_NOVERSION indicates that ICS version information cannot be verified. It occurs when the specified file does not contain the required INFOConnect RCDATA version information in its resource file. This may occur as the result of memory or disk corruption.

The user seeing this error should reinstall the offending product before retrying the action. If the error still occurs, contact the component's vendor.

IC_ERROR_NOXMTMEM (Value 308)

Internal error: no transmit memory. Free some memory and try again.

Level: Severe

For ICS DosLink applications, the IC_ERROR_NOXMTMEM internal error is returned when an attempt to allocate or access an ICS memory block fails. It occurs in low memory conditions.

The user seeing this error should free some memory before continuing.

IC_ERROR_OLDVERSION (Value 614)

This component requires obsolete version <number>. Contact the component's vendor for a software upgrade.

Level: Severe

IC_ERROR_OLDVERSION indicates that the component cannot be executed. It occurs when an application requires an older version of ICS.

The user seeing this error should contact the component's vendor to obtain an updated version of the component.
Errors and Results

IC_ERROR_PATHBUSY (Value 510)

Path <path ID> is currently active. Multiple instances of this path are not allowed.

Level: Severe

IC_ERROR_PATHBUSY indicates that the given path cannot be opened. It occurs when an attempt is made to open a second session over a non-multiplexing path.

The user seeing this error should close the active session before attempting to open another session using that path.

IC_ERROR_PATHID_EXISTS (Value 908)

Path ID already exists. Use a different ID or rename the existing path.

Level: Severe

IC_ERROR_PATHID_EXISTS indicates that a path ID already exists. It occurs when an attempt is made to add a path with a path ID that is already assigned.

The user seeing this error should either use a different ID or rename the existing path ID.

IC_ERROR_PICHANNELINUSE (Value 504)

Channel <identifier> in use. Not sharable between <library> and <library> external interface libraries.

Level: Severe

IC_ERROR_PICHANNELINUSE indicates that the given channel cannot be used by both of the given external interface libraries simultaneously. It occurs from IcOpenSession when the user attempts to use a single multiplexing service library with two different external interface libraries over the same channel.

The user seeing this error should close the active session before opening a session over the other path.

IC_ERROR_PIVERSION (Value 602)

<name> is not a valid INFOConnect external interface library. Library not loaded. Try reinstalling this product.

Level: Severe
IC_ERROR_PIVERSION indicates that the given file cannot be loaded as an external interface library. It occurs when the specified file does not properly identify itself as an INFOConnect EIL.

The user seeing this error should reinstall the offending product before trying to open the session. If the error still occurs, contact the component’s vendor.

IC_ERROR_PMCHANNELINUSE (Value 505)

Channel <identifier> in use. Not sharable between <library> and <library> service libraries.

Level: Severe

IC_ERROR_PMCHANNELINUSE indicates that the given channel cannot be used by both of the given service libraries simultaneously. It occurs from IcOpenSession when the user attempts to use a single multiplexing service library with two different external interface libraries over the same channel.

The user seeing this error should close the active session before opening a session over the other path.

IC_ERROR_PMVERSION (Value 601)

<name> is not a valid INFOConnect service library. Library not loaded. Try reinstalling this product.

Level: Severe

IC_ERROR_PMVERSION indicates that the given file cannot be loaded as a service library. It occurs when the specified file does not properly identify itself as an INFOConnect SL.

The user seeing this error should reinstall the offending product before trying to open the session. If the error still occurs, contact the component’s vendor.

IC_ERROR_QUEUEFULL (Value 304)

Queue full.

Level: Severe

For ICS DosLink applications, the IC_ERROR_QUEUEFULL internal error indicates that a message cannot be posted to a DosLink application. It occurs when the DosLink.386 driver’s queue is full.
Errors and Results

The user seeing this error should give control to the application so that some messages may be delivered.

IC_ERROR_RCV_BUSY (Value 10)

Station is still receiving. Request ignored.

Level: Severe

IC_ERROR_RCV_BUSY indicates that a receive request is still outstanding. It occurs when a second request to receive data is made before the first one completes. The accessory should wait for a receive-done or a receive-error type message before requesting more data.

The user seeing this error should wait until the accessory receives data for the outstanding request before making another receive request.

IC_ERROR_REOPEN (Value 2)

Internal Error.
Attempt to re-open external interface library.
Contact the component vendor for further instruction.

Level: Severe

IC_ERROR_REOPEN indicates an internal error. It occurs when an attempt is made to reopen a communications device.

The user should contact the component’s vendor.
Errors and Results

IC_ERROR_SERVICE_NOT_AVAILABLE (Value 1001)

Unavailable service requested: <service name>. Verify the service name with the component's documentation.

Level: Severe

IC_ERROR_SERVICE_NOT_AVAILABLE indicates that the request for the given service cannot be fulfilled. It occurs when a request is made for a service that is not supported.

The user seeing this error should verify that the service name is correct by referring to the component's documentation.

IC_ERROR_SHELL_ACTIVE (Value 103)

An INFOConnect Shell is already active.
You cannot run multiple shells.

Level: Termination

The IC_ERROR_SHELL_ACTIVE error indicates that an attempt has been made to execute two INFOConnect Shell applications. It occurs when an accessory tries to register itself as the INFOConnect Shell through IcInitShell and an INFOConnect Shell is already running. Only one ICS Shell may be active at a time. The second must terminate.

The user seeing this error should terminate the offending application.

IC_ERROR_SIZE_EXCEEDED (Value 904)

ID length limited to %d characters. Correct the ID and try again.

Level: Severe

IC_ERROR_SIZE_EXCEEDED indicates that a ID is too big. It occurs when an ID exceeds IC_MAX***IDLEN.

The user seeing this error should correct the ID and try the action again.
Errors and Results

IC_ERROR_SPACE_PRESENT (Value 905)

Space not allowed in ID. Correct the ID and try again.

Level: Severe

IC_ERROR_SPACE_PRESENT indicates that an ID erroneously contains a space. ICS does not allow spaces in IDs.

The user seeing this error should correct the ID and try the action again.

IC_ERROR_TERMINATECLEAR (Value 104)

A request has been made to clear this session.

Level: Termination

IC_ERROR_TERMINATECLEAR simply notifies an application that a communication session is being cleared. It occurs when the user chooses the Clear button from the INFOConnect user interface. The application has a chance to intercept this error and perform its termination routine before allowing the session to terminate. If the default error procedure is called, the session will close automatically. Unless INFOConnect is being executed in Debug mode, the associated error text will not be displayed by the default error procedure.

The user seeing this error should choose OK on the default error dialog to allow the session to close. The user will not see this error unless a -d appears as a command line parameter to INFOConnect.
IC_ERROR_TERMINATE_EXIT (Value 105)

A request has been made to close this session so INFOConnect can exit.

Level: Termination

IC_ERROR_TERMINATE_EXIT simply notifies the application that a communication session is being terminated because the user is closing INFOConnect Connectivity Services. The application has a chance to intercept this error and perform its termination routine before allowing the session to terminate. If the default error procedure is called, the session will close automatically. Unless INFOConnect is being executed in Debug mode, the associated error text will not be displayed by the default error procedure.

The user seeing this error should choose OK on the default error dialog to allow the session to close. The user will not see this error unless a -d appears as a command line parameter to INFOConnect.

IC_ERROR_TERMINATE_NOMSG (Value 0)

A request has been made to unconditionally terminate this session.

Level: Termination

IC_ERROR_TERMINATE_NOMSG simply requests that a communication session be unconditionally terminated. This is the error that is generated by the Local EIL when one half of the connected communications session is closed, and also by the IcTELNET SL if the TCP socket is closed. If the default error procedure is called, the session will close automatically. Unless INFOConnect is being executed in Debug mode, the associated error text will not be displayed by the default error procedure.

The user seeing this error should choose OK on the default error dialog to allow the session to close. The user will not see this error unless a -d appears as a command line parameter to INFOConnect.
Errors and Results

IC_ERROR_TERMINATE_SHUTDOWN (Value 106)

A request has been made to close this session so workstation can shutdown.

Level: Termination

IC_ERROR_TERMINATE_SHUTDOWN simply notifies the application that a communication session is being terminated because the user is closing Windows. The application has a chance to intercept this error and perform its termination routine before allowing the session to terminate. If the default error procedure is called, the session will close automatically. Unless INFOConnect is being executed in Debug mode, the associated error text will not be displayed by the default error procedure.

The user seeing this error should choose OK on the default error dialog to allow the session to close. The user will not see this error unless a -d appears as a command line parameter to INFOConnect.

IC_ERROR_TILDE_PRESENT (Value 907)

Tilde(~) not allowed in ID. Correct the ID and try again.

Level: Severe

IC_ERROR_TILDE_PRESENT indicates that an ID erroneously contains a tilde (~). ICS does not allow tildes in IDs.

The user seeing this error should correct the ID and try the action again.

IC_ERROR_TIMERS (Value 1)

Too Many Timers. Terminate some timers and retry.

Level: Severe

IC_ERROR_TIMERS indicates that a Windows timer cannot be started. It occurs when an attempt is made to start a Windows timer and the maximum number of timers has already been reached.

The user seeing this error should terminate some Windows applications that are using the timer resource and try the action again.
Errors and Results

IC_ERROR_TRUNCATED (Value 2002)

Buffer too small. String truncated.
Level: Informational

IC_ERROR_TRUNCATED result indicates that the output data has been truncated.
The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

IC_ERROR_UNIMPLEMENTED (Value 2012)

The requested function is not implemented.
Level: Informational

The IC_ERROR_UNIMPLEMENTED result is returned from function stubs that have not yet been implemented. This error should not occur in the released version of a product.
The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

IC_ERROR_UNKNOWN (Value 1000)

Unknown error encountered. Contact the component's vendor for more information.
Level: Severe

IC_ERROR_UNKNOWN indicates an unknown error. Developer's should attempt to use more descriptive errors.
The user seeing this error should contact the component's vendor.
Errors and Results

IC_ERROR_UNKNOWN_COMMAND (Value 2010)

Unknown command.

Level: Informational

The IC_ERROR_UNKNOWN_COMMAND result indicates that a command parameter is unknown.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

IC_ERROR_UNKNOWN_PARAMETER (Value 2009)

Unknown parameter.

Level: Informational

The IC_ERROR_UNKNOWN_PARAMETER result indicates that a parameter value is unknown.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

IC_ERROR_UNKNOWN_TABLE (Value 2011)

Unknown table.

Level: Informational

The IC_ERROR_UNKNOWN_TABLE result indicates that a table parameter is unknown.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.
IC_ERROR_UNOPENEDSESSION (Value 506)

Attempt to use unopened session. Verify the path configuration and clear the session, if necessary.

Level: Severe

IC_ERROR_UNOPENEDSESSION indicates that the session is not yet available for communication. It occurs when an application attempts to use a session handle that either does not exist or that has not yet finished establishing. See the IC_SessionEstablished message type (or the E_IC_SESSION_EST event type) for more information.

The user seeing this error may need to clear the session and verify that the path configuration is correct before reopening the session.

IC_ERROR_UPGRADE_WAIT (Value 613)

Library <name> is waiting for configuration data upgrade. Select Configure from the Configure Packages window.

Level: Severe

IC_ERROR_UPGRADE_WAIT indicates that the quick configuration accessory has not performed the data upgrade for a library whose data record format has changed. It may occur when quick configuration is abnormally terminated.

The user seeing this error should run quick configuration for the "Incomplete" packages to force the data upgrade to occur.

IC_ERROR_WRONGVERSION (Value 604)

Current version of INFOConnect does not support this version of <name>.
Library not loaded. Upgrade the necessary software.

Level: Severe

IC_ERROR_WRONGVERSION indicates that the given library cannot be executed. It occurs when the version of the specified ICS library is not supported by the current running version of ICS.

The user seeing this error should either update the library software or the ICS software.

IC_ERROR_XMT_BUSY (Value 6)

Station is still transmitting. Request ignored.
Errors and Results

Level: Severe

**IC_ERROR_XMT_BUSY** indicates that a transmit request is still outstanding. It occurs when a second request to transmit data is made before the first one completes. The accessory should wait for a transmit-done or a transmit-error type message before retransmitting.

The user seeing this error should wait until the accessory transmits data for the outstanding request before making another transmit request.

**IC_IGNORE** (Value 2007)

This request is being ignored at this time.

Level: Informational

The **IC_IGNORE** result indicates that a request is being ignored. The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

**IC_INCOMPLETE** (Value 2006)

The request cannot be completed at this time.

Level: Informational

The **IC_INCOMPLETE** result indicates that a request could not be completed. The request may be completed at a later time. See the **IC_COMPLETE** informational result.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.
**IC_INFO_QEVENT (Value 320)**

Message Queued.

Level: **Informational**

The **IC_INFO_QEVENT** result occurs when querying the ICS DosLink **IcNextEvent** API with the **IC_NEXTEVENT_CHECK** flag and indicates that at least one event is queued for the session.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

**IC_OK (Value 0)**

No Error.

Level: **Informational**

The **IC_OK** result indicates a successful completion of the request.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.

**IC_VERIFY_OK (Value 2005)**

The requested session may be successfully opened.

Level: **Informational**

The **IC_VERIFY_OK** result indicates that a verify action completed successfully.

The user seeing this result may wish to log it for future reference. Otherwise, the result may be ignored.
Errors and Results

ICS Standard Configurator Errors

The following error results are common/general errors defined for the ICS standard configurator. They may be returned as the result of a procedure call or with an error event (IC_Error, IC_RecError, and so forth under MS-Windows or E_IC_ERROR, and so forth under XVT). These errors have the IC_RESULT_CONTEXT_CFG context.

IC_CFG_ALREADY_ACTIVE (Value 141)

Init Config already active.

Level: Warning

IC_CFG_ALREADY_ACTIVE may occur in the ICS 2.02 release when a configuration application attempts to initialize the configuration API twice.

The user seeing this error should contact the component's vendor.

IC_CFG_BIT_FIELD (Value 134)

Bit field unsupported. Contact the component's vendor for more information.

Level: Severe

IC_CFG_BIT_FIELD indicates that an invalid action is being attempted on a bit field. See the IcGetField, IcSetField, IcGetKey, and IcSetKey functions. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_DATA_MISMATCH (Value 113)

Data format mismatch. Contact the component's vendor for more information.

Level: Severe

IC_CFG_DATA_MISMATCH occurs when a given IC_DATA_INFO.Length or IC_DATA_INFO.TableRevisionNum does not match those of the selected object. It will also occur if the size of requested data exceeds the IC_DATA_INFO.DataLength for the selected object. See the IcGetLibData, IcSetLibData, and IcCopyLibConfig functions.

The user seeing this error should contact the component's vendor.
IC_CFG_DATA_TRUNCATED (Value 133)

Data record truncated.

Level: Warning

IC_CFG_DATA_TRUNCATED indicates that the retrieved data record was truncated. It occurs when an attempt is made to retrieve a configuration data record and the output buffer parameter is not big enough to hold the context string. See IcGetLibDataInfo for information on retrieving information about the configuration data record.

The user seeing this error should contact the component's vendor for further information.

IC_CFG_DELETE_INUSE (Value 143)

Request to delete item rejected. You must first delete all references to the item. Still in use by <identifier>.

Level: Severe

IC_CFG_DELETE_INUSE occurs if an attempt is made to delete a configuration object that is configured as part of another configuration object. For example, a template cannot be deleted if a path is configured with it; a library or channel cannot be deleted if a path or template is configured using it; and a library's path configuration cannot be deleted if an ICS path is configured with it.

The user seeing this error should modify the configuration so that the configuration does not access the configuration object before deleting it.

IC_CFG_DIFFERENT_ACTIVE (Value 140)

Init Config of different version already active. Contact the component's vendor for more information.

Level: Severe

IC_CFG_DIFFERENT_ACTIVE may occur in the ICS 2.02 release when a configuration application attempts to initialize the configuration API twice.

The user seeing this error should contact the component's vendor.

IC_CFG_INFO_EXCESS (Value 132)

Excess info requested.
Errors and Results

Level: **Warning**

IC_CFG_INFO_EXCESS is currently not referenced.

**IC_CFG_INFO_IMPOSSIBLE** (Value 127)

Retrieval of informational data is impossible. Contact the component's vendor for more information.

Level: **Severe**

IC_CFG_INFO_IMPOSSIBLE occurs when a request for configuration information (IC_TABLE_INFO, IC_DATA_INFO, IC_KEY_INFO, IC_FIELD_INFO) is made and the length parameter is less than the size of the appropriate record for any version of ICS. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

**IC_CFG_INFO_TRUNCATED** (Value 131)

Additional info available.

Level: **Warning**

IC_CFG_INFO_TRUNCATED is currently not referenced.

**IC_CFG_INTERNAL_ERROR** (Value 100)

Internal Configuration API Error. Contact the component's vendor for more information.

Level: **Severe**

IC_CFG_INTERNAL_ERROR is an internal error with the Configuration Accessory API.

The user seeing this error should contact the configuration accessory's vendor for further instruction.

**IC_CFG_INVALID_DATABASE** (Value 160)

Invalid HIC_DATABASE. Contact the component's vendor for more information.

Level: **Severe**
IC_CFG_INVALID_DATABASE indicates that an HIC_DATABASE parameter is invalid. This error occurs when attempting to access a database that has never been opened or that has already been closed. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

**IC_CFG_INVALID_DB (Value 105)**

Invalid IC_DB parameter. Contact the component's vendor for more information.

**Level: Severe**

IC_CFG_INVALID_DB indicates that an IC_DB parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

**IC_CFG_INVALID_DBMODE (Value 106)**

Invalid IC_DB_MODE parameter. Contact the component's vendor for more information.

**Level: Severe**

IC_CFG_INVALID_DBMODE indicates that an IC_DB_MODE parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.
Errors and Results

IC_CFG_INVALID_FIELD (Value 109)

Invalid Field number. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_FIELD occurs when a reference is made to a non-existing field number, non-existing field name, or a non-existing IC_FIELD_REVISIONNUM. Check the component's .HIC include file for currently defined field numbers. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_FIELD_TYPE (Value 111)

Invalid field type parameter. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_FIELD_TYPE indicates that the field type parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_HANDLE (Value 103)

Invalid HIC_CONFIG. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_HANDLE indicates that the handle parameter is invalid. This occurs when the configuration object has never been opened, a severe error occurred during the open, or the object was already closed. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.
Errors and Results

IC_CFG_INVALID_HWND (Value 161)

Invalid configuration window handle. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_HWND indicates that the window handle parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_KEY (Value 108)

Invalid Key number. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_KEY occurs when a reference is made to a non-existing key number, or to a table that has no keys or IC_KEY_SERIALNUM defined. Check the component's .HIC include file for currently defined key numbers. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_LIBRARY (Value 104)

Attempt to select library failed. Verify the library ID and try again.

Level: Severe

IC_CFG_INVALID_LIBRARY indicates that a library's configuration cannot be accessed. The error occurs when an attempt is made to access an invalid library ID.

The user seeing this error should verify that the library ID is correct.
Errors and Results

IC_CFG_INVALID_POSITION (Value 112)

Invalid IC.Position parameter. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_POSITION indicates that the position parameter is invalid. This error occurs when IcPositionConfig receives an invalid IC.Position parameter, or receives the IC_POS_NEXTDUP, IC_POS_NEXT, or IC_POS_PREVIOUS parameter when no configuration object is currently selected. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_PROPERTY (Value 116)

Unsupported property parameter. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_PROPERTY indicates that the property parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_SIZE (Value 114)

Unsupported field size. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_SIZE occurs when an attempt is made to get (or set) a field or key with a variable when the conversion between the variable and the field/key is unsupported. For example, trying to get a 2 byte integer into a 1 byte variable will result in this error. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_TABLE (Value 107)

Invalid Table parameter. Contact the component's vendor for more information.

Level: Severe
IC_CFG_INVALID_TABLE indicates that the table parameter does not reference a valid configuration table for the currently selected component. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_TABLE_TYPE (Value 110)

Invalid table type parameter. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_TABLE_TYPE indicates that the table type parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_INVALID_TEMPLATE (Value 162)

Path config contains invalid Path Template. Contact the component's vendor for more information.

Level: Severe

IC_CFG_INVALID_TEMPLATE indicates that the table type parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.
IC_CFG_INVALID_TYPE (Value 115)

Unsupported field type. Contact the component’s vendor for more information.

Level: Severe

IC_CFG_INVALID_TYPE occurs when an attempt is made to get (or set) a field or key with an incompatible type. For example, trying to get a IC_FT_INTEGER of a field defined as IC_FT_CHAR will result in this error. This error should not occur in the released version of a product.

The user seeing this error should contact the component’s vendor.

IC_CFG_INVALID_TYPE_SIZE (Value 135)

Unsupported field type/size. Contact the component’s vendor for more information.

Level: Severe

IC_CFG_INVALID_TYPE_SIZE occurs when an attempt is made to get (or set) a field or key with an incompatible field size/field type combinations. This error should not occur in the released version of a product.

The user seeing this error should contact the component’s vendor.

IC_CFG_MISMATCH_DATA (Value 126)

Mismatch data format. Contact the component’s vendor for more information.

Level: Severe

IC_CFG_MISMATCH_DATA is currently not referenced.

IC_CFG_NAME_TRUNCATED (Value 130)

Retrieved name truncated.

Level: Warning

IC_CFG_NAME_TRUNCATED indicates that the retrieved field name (IcGetFieldName) or table name (IcGetLibTableName) was truncated. It occurs when an attempt is made to retrieve a field name or table name and the output buffer parameter is not big enough to hold the name.
Errors and Results

**IC_CFG_NEW_DATA (Value 128)**

New record created.

Level: Warning

**IC_CFG_NEW_DATA** indicates that a new configuration data record was created from the default configuration data. It occurs from those configuration functions that automatically create new records when the database is read/write and the requested record does not currently exist. See **IcFindNewConfig**.

The user seeing this error should modify the default configuration data appropriately.

**IC_CFG_NO_DATA_MEMORY (Value 136)**

No memory to load configuration data. Free some memory and try again.

Level: Severe

**IC_CFG_NO_DATA_MEMORY** indicates that a configuration object data buffer cannot be allocated. It occurs in low memory conditions.

The user seeing this error should free some memory and retry the action.

**IC_CFG_NO_HCFG_MEMORY (Value 139)**

No memory to open config session. Close some ICS configuration applications and try again.

Level: Severe

**IC_CFG_NO_HCFG_MEMORY** occurs when a configuration object cannot be allocated. It occurs in low memory conditions.

The user seeing this error should close one or more ICS configuration applications and try the action again.
Errors and Results

**IC_CFG_NO_HDB_MEMORY (Value 138)**

No memory to open config database. Close some ICS configuration applications and try again.

Level: Severe

IC_CFG_NO_HDB_MEMORY occurs when the database configuration object cannot be allocated. It occurs in low memory conditions.

The user seeing this error should close one or more ICS configuration applications and try the action again.

**IC_CFG_NO_HLIB_MEMORY (Value 162)**

No memory to open config library. Close some ICS configuration applications and try again.

Level: Severe

IC_CFG_NO_HLIB_MEMORY occurs when the database configuration object cannot be allocated. It occurs in low memory conditions.

The user seeing this error should close one or more ICS configuration applications and try the action again.

**IC_CFG_NO_INFO_MEMORY (Value 137)**

No memory to load configuration info. Free some memory and try again.

Level: Severe

IC_CFG_NO_INFO_MEMORY occurs when an attempt to allocate a configuration definition buffer fails. It occurs in low memory conditions.

The user seeing this error should free some memory and try the action again.
Errors and Results

IC_CFG_NO_INIT (Value 102)

Application never IcInitConfig. Contact the component's vendor for more information.

Level: Severe

IC_CFG_NO_INIT may occur in the ICS 2.02 release when a configuration application attempts to use the configuration API before it initializes it.

The user seeing this error should contact the component's vendor.

IC_CFG_NOT_FOUND (Value 125)

Configuration data not found. Verify the configuration.

Level: Severe

IC_CFG_NOT_FOUND indicates that a configuration record could not be found. It occurs when either the requested record is missing, or from IcPositionConfig when there are no more entries at which to position.

The user seeing this error should verify that the configuration is correct.

IC_CFG_NOT_IMPLEMENTED (Value 101)

Configuration API not implemented. Contact the component's vendor for more information.

Level: Severe

IC_CFG_NOT_IMPLEMENTED occurs when requesting service from configuration API that has not yet been implemented. All functions will be implemented as documented in a future ICS release. Developers should code accordingly. Therefore, this error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.
IC_CFG_STILL_ACTIVE (Value 142)

Init Config is still active.

Level: Warning

IC_CFG_STILL_ACTIVE indicates that configuration objects are still open and active. This error occurs when a configuration application closes a configuration session before closing the active configuration objects. The configuration objects remain active.

The user seeing this error should complete the configuration task.

IC_CFG_UNKNOWN_COMPONENT (Value 119)

Unknown component. Contact the component's vendor for more information.

Level: Severe

IC_CFG_UNKNOWN_COMPONENT is currently not referenced.

IC_CFG_UNKNOWN_FIELDTYPE (Value 122)

Unknown field type. Contact the component's vendor for more information.

Level: Severe

IC_CFG_UNKNOWN_FIELDTYPE is currently not referenced.

IC_CFG_UNKNOWN_GENERIC (Value 121)

Unknown generic component. Contact the component's vendor for more information.

Level: Severe

IC_CFG_UNKNOWN_GENERIC is currently not referenced.
Errors and Results

IC_CFG_UNKNOWN_PROPERTY (Value 118)
Unknown property number. Contact the component's vendor for more information.

Level: Severe

IC_CFG_UNKNOWN_PROPERTY indicates that the property parameter is unsupported. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_UNKNOWN_ROLE (Value 117)
Unknown role parameter. Contact the component's vendor for more information.

Level: Severe

IC_CFG_UNKNOWN_ROLE indicates that the role parameter is invalid. This error should not occur in the released version of a product.

The user seeing this error should contact the component's vendor.

IC_CFG_UNKNOWN_SUPPLIER (Value 120)
Unknown supplier. Contact the component's vendor for more information.

Level: Severe

IC_CFG_UNKNOWN_SUPPLIER is currently not referenced.

IC_CFG_UNSAVED_DATA (Value 129)
Unsaved data discarded.

Level: Warning

IC_CFG_UNSAVED_DATA is currently not referenced.
Errors and Results

IC_CFG_WRONG_FIELDSIZE (Value 123)
Wrong field size. Contact the component's vendor for more information.
Level: Severe
IC_CFG_WRONG_FIELDSIZE is currently not referenced.

IC_CFG_WRONG_FIELDTYPE (Value 124)
Wrong field type. Contact the component's vendor for more information.
Level: Severe
IC_CFG_WRONG_FIELDTYPE is currently not referenced.
IcACOMS

IcACOMS is a multiplexing library that manages A Series COMS-specific communications protocol. The `icacoms.hic` include file defines the generic interface of the IcACOMS AIL.

IcACOMS Errors

The following error values are specific to the IcACOMS library. They are distinguished by the context associated with the context string `COMS_CONTEXTSTRING`, defined in the `icacoms.hic` include file. Include this file in an application that is coded to be aware of these specific errors.

**COMS_CHANNEL_ACTIVE** (Value 225)

Channel already active.

Level: Warning

The `COMS_CHANNEL_ACTIVE` warning occurs when an attempt is made to open a COMS channel that is already open. Each COMS channel can only be opened once.

The user seeing this error should not try to open the channel again.

**COMS_ERROR_ACTIVESESS** (Value 211)

Internal Error. `OpenSession` requested by an active session. Contact the component vendor for further instruction.

Level: Severe

`COMS_ERROR_ACTIVESESS` is an internal error that occurs when an attempt is made to re-open a COMS communication session that has not been properly closed.

The user seeing this error should contact the vendor for further instruction.
Errors and Results

COMS_ERROR_DUPLICATE (Value 214)

Selected CUSTOM PATH is already active.

Level: Severe

COMS_ERROR_DUPLICATE occurs when an attempt is made to establish a communication session on a COMS custom window that is already active.

The user seeing this error should not try to open the second instance of the custom path. Only one instance of each custom path can be active at a time.

COMS_ERROR_INSERTCHANNEL (Value 216)

Memory error while accessing Channel List.
Free up some memory and try again.

Level: Severe

COMS_ERROR_INSERTCHANNEL indicates that an internal IcACOMS error has occurred, and usually signifies a memory error.

The user seeing this error should free some memory and try again.

COMS_ERROR_INSERTSESSION (Value 215)

Memory error while accessing Session List.
Free up some memory and try again.

Level: Severe

COMS_ERROR_INSERTSESSION indicates that an internal IcACOMS error has occurred, and usually signifies a memory error.

The user seeing this error should free some memory and try again.
COMS_ERROR_INSERTWINDOWS (Value 217)
Memory error while accessing COMS Windows List. Free up some memory and try again.
Level: Severe

COMS_ERROR_INSERTWINDOWS indicates that an internal IcACOMS error has occurred, and usually signifies a memory error.

The user seeing this error should free some memory and try again.

COMS_ERROR_MAXDIALOGS (Value 212)
Maximum COMS Dialogs Active. Close some and try again.
Level: Severe

COMS_ERROR_MAXDIALOGS occurs when the maximum number of COMS dialogs has been reached.

The user seeing this error should either close some dialogs and try again or open the session on a different COMS window.
Errors and Results

IcHLCNTS

IcHLCNTS is an external interface library that provides an interface to A Series Host Lan Connection (HLCN) Terminal Services (TS). The ichlcnts.hic include file defines the generic interface of the IcHLCNTS EIL.

The IcHLCNTS EIL also acts as the package's configuration library. The package configuration table is defined by ICHLCNTS_HOST_TABLENUM.

IcHLCNTS Errors

The following error values are specific to the IcHLCNTS external interface library. They are distinguished by the context associated with the context string HCLNTS_CONTEXTSTRING, defined in the ichlcnts.hic include file. Include this file in an application that is coded to be aware of these specific errors.

NTS_CONNECT_DENIED (Value 3)

HOST DENIED CONNECTION

Level: Severe

NTS_CONNECT_DENIED indicates that the A Series host rejected the connection of the terminal name configured in the path.

The user seeing this error should check the configuration and verify the terminal name with the MIS department.

If the user has configured this path to receive messages, then this error text appears as a message on the terminal screen. Otherwise, the error is returned as an error message to the application. See NTS_CONNECT_REJECTED.
Errors and Results

**NTS_CONNECT_FAILED** (Value 2)

CONNECT REQUEST FAILED

Level: Severe

**NTS_CONNECT_FAILED** indicates that the NetBIOS connection was lost while waiting for the host to connect. It occurs when IcHCLNTS receives a status message from NetBIOS indicating that the session is closed.

The user seeing this error should hit transmit to try to re-establish the session.

If the user has configured this path to receive messages, then this error text appears as a message on the terminal screen. Otherwise, the error is returned as an error message to the application.

**NTS_CONNECT_LOST** (Value 4)

CONNECTION HAS BEEN LOST

Level: Severe

**NTS_CONNECT_LOST** indicates that the NetBIOS connection to the host has been lost or the host terminated the session. It occurs when IcHCLNTS receives a status message from NetBIOS indicating that the session is closed.

The user seeing this error should hit transmit to try to re-establish the session.

If the user has configured this path to receive messages, then this error text appears as a message on the terminal screen. Otherwise, the error is returned as an error message to the application.
Errors and Results

**NTS_CONNECT_REJECTED** (Value 22)

Connection request denied: *(<host error>) <associated text>*
Transmit again to retry the connection attempt.

Level: Severe

**NTS_CONNECT_REJECTED** reports the error code received from the host, as well as the error text from the host. The error occurs when the A Series host rejects the connection attempt. It is returned to the application only if the path has not been configured to receive messages.

The user seeing this error should hit transmit to try to re-establish the session.

If the user has configured this path to receive messages, then the text associated with the **NTS_CONNECT_DENIED** error appears on the terminal screen followed by the error text from the host.

**NTS_CREDITS_EXCEEDED** (Value 23)


Level: Severe

**NTS_CREDITS_EXCEEDED** indicates that a message has been discarded. It occurs when the host has sent data that has exceeded the specified protocol limits.

The user seeing this error should trace both the configured path and the host path associated with the path's channel. The debug files should be sent to the component vendor for further action.

**NTS_MSG_OK** (Value 1)

.ok.

Level: Non-error

This terminal data message appears on the terminal screen if the user has configured the path for input edit and has issued a terminal options command (for example, ?+s, ?-i, etc.).

**NTS_NO_HOSTPATH** (Value 24)

HostPath *(<name>)* referenced by HLCNTS Channel *(<identifier>)* has not been configured.

Level: Severe
**Errors and Results**

**NTS_NO_HOSTPATH** occurs when an attempt is made to open the given host path that is associated with the given channel and that host path has not yet been configured.

The user seeing this error should configure the host path and try again.

**NTS_TERMINAL_ACTIVE (Value 21)**

Terminal `<name>` is already active. Unable to open terminal multiple times.

Level: Severe

**NTS_TERMINAL_ACTIVE** indicates that the open session request failed. It occurs when an attempt is made to open a second session using a terminal name that is already open.

The user seeing this error should not try to open a terminal session multiple times.
Errors and Results

IcLCW

IcLCW is a service library that, when used in conjunction with the IcxNS external interface library, provides LAN Connected Workstation-specific, value-added functionality. The iclcw.hic include file defines the generic interface of the IcLCW SL.

The IcLCW SL also acts as the package's configuration library. The package configuration table is defined by ICLCW_TEMPL_TABLENUM.

IcLCW Errors

There are no error results specific to the IcLCW service library.
IcLocal

IcLocal is an external interface library that manages data communications between two applications on the same system. The iclocal.h include file defines the generic interface of the IcLocal EIL.

IcLocal Errors

There are no error results specific to the IcLocal external interface library.
Errors and Results

IcMon

IcMon is a service library that maintains transaction-related information on a per-session basis. The generic component ID is IC_GENERIC_MON. The icmon.hic include file defines the generic interface of the IcMon SL.

IcMon Errors

The following error values are specific to the IcMon service library. They are distinguished by the context associated with the context string ICMON_CONTEXTSTRING, defined in the icmon.hic include file. Include this file in an application that is coded to be aware of these specific errors.

ICMON_ERR_KEYVALUE (Value 500)

Invalid key for monitor's configuration record.

Level: Severe

ICMON_ERR_KEYVALUE indicates that the ICMON_OPTIONSTABLE_KEY key is incorrect. It may occur as the result of memory or disk corruption.

The user seeing this error should reconfigure the monitor library. If the error still occurs, contact the component's vendor for further information.

ICMON_ERR_NODUPEOPTIONS (Value 502)

Only one Monitor Options record is allowed.

Level: Severe

ICMON_ERR_NODUPEOPTIONS indicates that the options table can only have one record. It occurs if an application attempts to add a second record to the Monitor's options table.

The user seeing this error should contact the component's vendor.
ICMON_ERR_RANGEVALUE (Value 501)

Invalid RANGE in monitor's configuration options. RANGE values must be in increasing order.

Level: Severe

ICMON_ERR_RANGEVALUE occurs when the transaction counters, denoted by fields ICMON_SESS_PREV_RT_RANGE1, ICMON_SESS_PREV_RT_RANGE2, and ICMON_SESS_PREV_RT_RANGE3 are not set in ascending order.

The user seeing this error should reconfigure the monitor library.
IcNBIOS

IcNBIOS is an external interface library that provides an interface to NetBIOS protocol stacks. The icnbios.hic include file defines the generic interface of the IcNBIOS EIL.

IcNBIOS Errors

The following error values are specific to the IcNBIOS external interface library. They are distinguished by the context associated with the context string NETBIOS_CONTEXTSTRING, defined in the icnbios.hic include file. Include this file in an application that is coded to be aware of these specific errors.

Note that the configurable Auto Connect feature of IcNBIOS alters the errors that will be reported. When this feature is enabled, the IcNBIOS EIL will automatically reconnect failed sessions. The error will be returned to the application with the informational error level. These are normally not displayed by the default error procedure.

NETBIOS_DUP_NAME (Value 4)

NetBIOS name already in use on the network.

Level: Termination

NETBIOS_DUP_NAME occurs when a request is made to claim a NetBIOS name that is already active on the network.

The user seeing this error must choose a different NetBIOS name before opening the session.

NETBIOS_ERRADATA (Value 8)

Error %x getting adapter data to retrieve the permanent node name.

Level: Severe

NETBIOS_ERRADATA reports the error number that occurs when adapter data, which contains the permanent node name, could not be retrieved.

The user seeing this error should supply a NetBIOS name in the path configuration and try again.
Errors and Results

**NETBIOS_ERR_ADD_NAME** (Value 5)

Error $%#2x$ adding NetBIOS name.
See NetBIOS documentation for more information.

Level: Termination

**NETBIOS_ERR_ADD_NAME** reports the error number that occurred when adding (claiming) a NetBIOS name on the network fails. This may be the result of an abnormal termination of INFOConnect.

The user seeing this error should reboot the machine.

**NETBIOS_ERR_CALL** (Value 7)

Error $%#2x$ on call. See NetBIOS documentation for more information.

Level: Severe

**NETBIOS_ERR_CALL** reports the error number that occurs when a call to the remote device cannot be performed.

The user seeing this error should refer to the NetBIOS documentation for more information on the given error number.

**NETBIOS_ERR_CONNECT** (Value 9)

NetBIOS call error $%#2x$. See NetBIOS documentation for more information.

Level: Termination or Informational

**NETBIOS_ERR_CONNECT** reports the error number that occurs when a call to the remote device completes in error.

This error is normally a terminate-type error. However, if the user has configured this path with auto connection and the error number is 05h, 12h, or 14h, then the error is returned to the application as an informational-type error and the NetBIOS call is attempted again.

The user seeing this error should close session and re-open it to try to reconnect.

**NETBIOS_ERR_DELETE_NAME** (Value 11)

Error $%#2x$ deleting NetBIOS name from the network.
See NetBIOS documentation for more information.

Level: Severe
Errors and Results

NETBIOS_ERR_DELETE_NAME reports the error number that occurs when deleting a NetBIOS name fails. The machine may have to be rebooted to reinitialize the local name table.

The user seeing this error should refer to the NetBIOS documentation for more information on the given error number.

NETBIOS_ERR_LISTEN (Value 6)

Error %#2x on listen.  
See NetBIOS documentation for more information.

Level: Severe

NETBIOS_ERR_LISTEN reports the error number that occurs when a listen for an incoming call was attempted. If the Auto Connect feature if IcNBIOS was enabled, the listen will be retried.

The user seeing this error should refer to the NetBIOS documentation for more information on the given error number.

NETBIOS_ERR_RECEIVE (Value 32)

Rcv error %#2x.  See NetBIOS documentation for more information.

Level: Severe or Informational

This error reports the error number that occurs when receiving a message.

NETBIOS_ERR_RECEIVE is normally a severe-type error (thus canceling the receive request). However, if the user has configured this path with auto connection and the error number is 0ah or 18h, then an informational-type error is reported to the application, the connection is closed, and an attempt is made to re-open the connection and continue the receive request.

The user seeing this error should refer to the NetBIOS documentation for more information on the given error number.

NETBIOS_ERR_RECEIVING (Value 22)

A receive is still pending.  Request ignored.

Level: Severe

NETBIOS_ERR_RECEIVING indicates that a receive request is still outstanding. It occurs when an attempt is made to issue a second receive request. Only one receive request may be outstanding at a time.
The user seeing this error should wait until the accessory receives data for the outstanding request before making another receive request.

**NETBIOS_ERR_SEND** (Value 33)

Xmt error (%#2x). See NetBIOS documentation for more information.

*Level:* **Severe** or **Warning**

**NETBIOS_ERR_SEND** reports the error number that occurs when transmitting a message.

This error is normally a severe-type error (thus canceling the transmit request). However, if the user has configured this path with auto connection and the error number is 0ah or 18h, then a warning-type error is reported to the application, the connection is closed, and an attempt is made to re-open the connection and continue the transmit request.

The user seeing this error should refer to the NetBIOS documentation for more information on the given error number.

**NETBIOS_ERR_SENDING** (Value 23)

A transmit is still pending. Request ignored.

*Level:* **Severe**

**NETBIOS_ERR_SENDING** indicates that a transmit request is still outstanding. It occurs when an attempt is made to issue a second transmit request. Only one transmit request may be outstanding at a time.

The user seeing this error should wait until the accessory transmits data for the outstanding request before making another transmit request.

**NETBIOS_INTERNAL** (Value 10)

ICNBIOS EIL internal error <number>.

Contact component vendor for more information.

*Level:* **Severe**

**NETBIOS_INTERNAL** reports an internal IcNBIOS error number.

The user seeing this error should report the IcNBIOS error number to the component's vendor.
Errors and Results

NETBIOS_NOT_FOUND (Value 3)

NetBIOS not found. Load NetBIOS before running Windows.

Level: Severe

NETBIOS_NOT_FOUND occurs when NetBIOS could not be found.

The user seeing this error should verify that NetBIOS is loaded before running Windows.

NETBIOS_XMT_BUSY (Value 21)

Station is still transmitting. Request to terminate transmit ignored.

Level: Severe

NETBIOS_XMT_BUSY occurs when a transmission is still in process and the application requested to cancel it. The IC_LCL_XMT request is ignored.

The user seeing this error should wait until the transmit request has completed.
IcTCP

The IcTCP external interface library provides generic TCP/IC socket access. The `ictcp.hic` include file defines the generic interface of the IcTCP EIL.

IcTCP Errors

There are no error results specific to the IcTCP external interface library.
Errors and Results

IcTELNET

The IcTELNET service library provides basic TELNET services over TCP/IP. This allows VT-type emulator to hosts which support TELNET (for example, U Series, 1100/2200 Series, A Series). The ictelnet.hic include file defines the generic interface of the IcTELNET SL.

The IcTELNET SL also acts as the package's configuration library. The package configuration table is defined by ICTEL_TEMPL_TABLENUM.

IcTELNET Errors

The following error values are specific to the IcTELNET service library. They are distinguished by the context associated with the context string TELNET_CONTEXTSTRING, defined in the ictelnet.hic include file. Include this file in an application that is coded to be aware of these specific errors.

TELNET_BAD_CONFIG (Value 12)

Internal Error. Contact the component vendor for more information.

Level: Severe

TELNET_BAD_CONFIG is an internal error that indicates that an error has occurred within the INFOConnect database. It may occur as the result of disk corruption.

The user seeing this error should contact the component vendor for more information.

TELNET_ERR_RECEIVING (Value 22)

A receive is still pending. Request ignored.

Level: Severe

TELNET_ERR_RECEIVING occurs when an attempt is made to issue a second receive request. Only one receive request may be outstanding at a time.

The user seeing this error should wait until the receive request has completed.

TELNET_ERR_SENDING (Value 23)

A transmit is still pending. Request ignored.

Level: Severe
**Errors and Results**

**TELNET_ERR_SENDING** occurs when an attempt is made to issue a second transmit request. Only one transmit request may be outstanding at a time.

The user seeing this error should wait until the transmit request has completed.

**TELNET_INTERNAL (Value 10)**

IcTELNET Service Library internal error `<number>`.
Contact component vendor for more information.

Level: Severe

**TELNET_INTERNAL** reports an internal IcTELNET error number.

The user seeing this error should report the IcTELNET error number to the component’s vendor.
Errors and Results

IcTrace

The IcTrace service library traces INFOConnect data communications calls and events and writes them to a trace file, trace.log, located in the DataDir directory. The ictrace.hic include file defines the generic interface of the IcTrace SL.

IcTrace Errors

There are no error results specific to the IcTrace service library.
IcTTY

IcTTY is an external interface library that manages a TTY connection through the computer's COM ports. The ictty.hic include file defines the generic interface of the IcTTY EIL.

IcTTY Errors

The following error values are specific to the IcTTY external interface library. They are distinguished by the context associated with the context string TTY_CONTEXTSTRING, defined in the ictty.hic include file. Include this file in an application that is coded to be aware of these specific errors.

The following errors prefixed by TTY_ERROR_... may occur during communication session establishment. They correspond to the results returned by MS-Windows if an error occurs while opening the Windows communication device.

TTY_ERROR_BAUDERROR (Value 8)

Baud rate is not supported. Reconfigure path and try again.

Level: Termination

TTY_ERROR_BAUDERROR indicates that the configured baud rate is unsupported.

The user seeing this error should reconfigure this path and try again.

TTY_ERROR_BYTEERROR (Value 7)

Invalid byte size specified. Reconfigure path and try again.

Level: Termination

TTY_ERROR_BYTEERROR indicates that the configured byte size is invalid.

The user seeing this error should reconfigure this path and try again.
Errors and Results

TTY_ERROR_DEFPARAM (Value 5)

Default parameters are bad. Verify configuration.

Level: Termination

TTY_ERROR_DEFPARAM indicates that the default parameters are invalid.

The user seeing this error should reconfigure this path, verify the Windows communications port configuration, and try again.

TTY_ERROR_DIALABORTED (Value 11)

User Aborted Autodialing.

Level: Termination

TTY_ERROR_DIALABORTED indicates that the user aborted the auto dialing feature of the IcTTY EIL. The session will not be opened.

TTY_ERROR_NOPORT (Value 1)

Com port does not exist. Reconfigure and try again.

Level: Termination

TTY_ERROR_NOPORT indicates that the communication ID is invalid or unsupported.

The user seeing this error should reconfigure this path, verify the Windows communications port configuration, and try again.

TTY_ERROR_NOQs (Value 4)

Unable to allocate I/O queues. Free up some memory and try again.

Level: Termination

TTY_ERROR_NOQs indicates that there is not enough memory to allocate the input/output queues. It occurs in low memory conditions.

The user seeing this error should free some memory and try again.

TTY_ERROR_NOTIMER (Value 10)

Dialing timer could not be started.

Level: Termination
TTY_ERROR_NOTIMER occurs when an attempt to start the auto dialing Windows timer fails. Dialing cannot continue, and the session will not be opened.

The user seeing this error should terminate some Windows applications that are using the timer resource and try again.

TTY_ERROR_NOTOPEN (Value 3)

Com port is not open. Verify Windows communication port configuration.

Level: Termination

TTY_ERROR_NOTOPEN indicates that the communication device could not be opened.

The user seeing this error should verify that the Windows communication port configuration is correct.

TTY_ERROR_OPEN (Value 2)

Device is already open. Verify that another application is not using the communications port.

Level: Termination

TTY_ERROR_OPEN indicates that the communication device is already open.

The user seeing this error should verify that another application is not using the communications port.
Errors and Results

TTY_ERROR_UNAVAILPORT (Value 6)

Com port is not available. Verify communications hardware.

Level: Termination

TTY_ERROR_UNAVAILPORT indicates that the device hardware is not available.

The user seeing this error should verify that the communications hardware is correctly installed and operational.

TTY_ERROR_UNKNOWN (Value 9)

Unknown status returned by Windows.

Level: Termination

TTY_ERROR_UNKNOWN indicates that an unknown error result was returned by MS-Windows.

The user seeing this error should contact the component's vendor.

TTY_LCLERROR_FAILED (Value 40)

The communications port could not be set into Local mode. Verify handshaking configuration.

Level: Termination

TTY_LCLERROR_FAILED indicates that the previous request to set the communications port into local mode did not succeed.

The user seeing this error should verify the handshaking configuration.

TTY_RCVERROR_FAILED (Value 22)

The communications port could not be set into Receive mode. Verify handshaking configuration.

Level: Termination

TTY_RCVERROR_FAILED indicates that the previous request to set the communications port into Receive mode did not succeed.

The user seeing this error should verify the handshaking configuration.
Errors and Results

TTY_RCVERROR_FRAME (Value 21)

The hardware detects a framing error.
Check hardware and verify hardware configuration.

Level: Severe
TTY_RCVERROR_FRAME occurs when the hardware detects a framing error.
The user seeing this error should verify the hardware and the hardware configuration.

TTY_RCVERROR_OVERRUN (Value 20)

A receive overrun error has occurred, data has been lost.
Contact the component vendor for further instruction.

Level: Severe
TTY_RCVERROR_OVERRUN occurs when data in the receive buffer is not read before more data arrives.
The user seeing this error should contact the component's vendor.

TTY_XMTERRO error CTSTO (Value 30)

Clear-to-send timeout. Check wiring and verify configuration.

Level: Severe
TTY_XMTERRO error CTSTO occurs when the Clear-to-send signal times out while trying to transmit.
The user seeing this error should verify the wiring and the configuration.

TTY_XMTERRO error DSRTO (Value 31)

Data-set-ready timeout. Check wiring and verify configuration.

Level: Severe
TTY_XMTERRO error DSRTO occurs when the Data-set-ready signal times out while trying to transmit.
The user seeing this error should verify the wiring and the configuration.

TTY_XMTERRO error RLSDTO (Value 32)

Receive-line-signal-detect timeout. Check wiring and verify configuration.
Errors and Results

TTY_XMTERROR_RLSDTO occurs when the Receive-line-signal-detect signal times out while trying to transmit.

The user seeing this error should verify the wiring and the configuration.

TTY_XMTERROR_TRANSMITTING (Value 34)

Station is still transmitting. Request ignored.

TTY_XMTERROR_TRANSMITTING indicates that the previous request to transmit has not completed. This request is rejected. The application/accessory should wait for a transmit-done or a transmit-error type event/message before retransmitting.

The user seeing this error should contact the component's vendor.

TTY_XMTERROR_TXFULL (Value 33)

The transmit queue is full while trying to queue a character. Contact the application vendor for further information.

TTY_XMTERROR_TXFULL occurs when data cannot be queued because the transmit queue is full.

The user seeing this error should contact the application's vendor.
IcXNS

IcXNS is an external interface library that allows access to network nodes on a Novell® LAN. The icxns.hic include file defines the generic interface of the IcXNS EIL.

IcXNS Errors

The following error values are specific to the IcXNS external interface library. They are distinguished by the context associated with the context string `XNS_CONTEXTSTRING`, which is defined in the icxns.hic include file. Include icxns.hic and dcdevice.hic in an application that is coded to be aware of these specific errors.

**DCDEV_BADDEVICE** (Value 1003)

Internal Error.
Invalid XNS device driver (XNSCOM.SYS).

Level: Severe

**DCDEV_BADDEVICE** occurs when the version of the installed XNSCOM.SYS device is not a valid XNS device.

The user seeing this error should reinstall the XNS device driver.

**DCDEV_NO_CHANNEL** (Value 1006)

No channel available. Increase /Sn parameter of XNSCOM.SYS device in config.sys or close an active session.

Level: Severe

**DCDEV_NO_CHANNEL** occurs when no channel is available.

The user seeing this error should increase the /Sn parameter for the XNS device driver in CONFIG.SYS and reboot the machine or close an active XNS session.
Errors and Results

DCDEV_NO_DEVICE (Value 1001)

Unable to open device. Verify that device=<path>XNSCOM.SYS is present in CONFIG.SYS.

Level: Severe

DCDEV_NO_DEVICE occurs when the required data communications device could not be opened.

The user seeing this error should verify that the device statement for the XNS device driver in CONFIG.SYS is correct.

DCDEV_NO_DRIVER (Value 1005)

XNSCOM.SYS requires IPX.COM to be loaded.

Level: Severe

DCDEV_NO_DRIVER occurs when IPX.COM is not loaded. IPX.COM must be loaded before loading Windows.

The user seeing this error should load IPX before loading Windows.

DCDEV_NOT_DEVICE (Value 1002)

Unable to verify device. Verify that CONFIG.SYS references the correct version of XNSCOM.SYS.

Level: Severe

DCDEV_NOT_DEVICE occurs when the required device could not be verified.

The user seeing this error should verify that CONFIG.SYS references the correct version of XNSCOM.SYS and reboot if necessary.
Errors and Results

DCDEV_OLD_DEVICE (Value 1004)

Old XNS device driver (XNSCOM.SYS).

Level: Warning or Severe

DCDEV_OLD_DEVICE occurs when the version of the installed XNSCOM.SYS device is older than the IcXNS.DLL library. If the library can continue, this is a warning type message; otherwise, it is severe.

The user seeing this error should verify that CONFIG.SYS references the correct version of XNSCOM.SYS and reboot if necessary.

DCDEV_READ_ERROR (Value 1020)

Internal Read error.
Contact the component vendor for further instruction.

Level: Severe

DCDEV_READ_ERROR indicates that the data communications device could not read data.

The user seeing this error should contact the component's vendor.

DCDEV_WRITE_ERROR (Value 1021)

Internal Write error.
Contact the component vendor for further instruction.

Level: Severe

DCDEV_WRITE_ERROR indicates that the data communications device could not write data.

The user seeing this error should contact the component's vendor.
Errors and Results

DCDEV_WRITE_INCOMPLETE (Value 1022)

Write incomplete. Verify that the /b parameter of XNSCOM.SYS device in config.sys matches the application's suggested value.

Level: Severe

DCDEV_WRITE_INCOMPLETE indicates that the data communications device could not complete writing data.

The user seeing this error should verify that the /b parameter for the XNS device driver in CONFIG.SYS matches the application's suggested value.

XNS_ADDRESS_ERROR (Value 701)

LAN terminal address error.

Level: Severe

XNS_ADDRESS_ERROR occurs when the LAN terminal address is in error.

The user seeing this error should verify the configured address.

XNS_SOCKET_ERROR (Value 702)

Same socket already open.

Level: Severe

XNS_SOCKET_ERROR occurs when the configured socket is already open.

The user seeing this error should verify that the socket configuration is correct.
Glossary

A

AAPI
See Accessory Application Programming Interface.

accessory
An ICS application that can be invoked and controlled by other ICS applications. Accessories are written to be useful in building more sophisticated products. An accessory adheres to the rules outlined in Section 6 of this manual.

Accessory Application Programming Interface (AAPI)
The interface available to INFOConnect applications and accessories. The AAPI defines a collection of services for sending and receiving data across a data communications connection in a transport-independent manner.

accessory ID
See ID.

AIL
See application interface library.

aliasing (channel and session)
The ICS Manager uses channel identifiers in the form of HIC_CHANNELs and session identifiers in the form of HIC_SESSIONs. Libraries must use these identifiers, or handles, when calling the ICS Manager API as needed. The library may create an alias for these identifiers by assigning a value that uniquely identifies the channel or session in the IcLibOpenChannel or IcLibOpenSession procedure. In this case, the library will receive this value on all calls from the ICS Manager. Otherwise, the library receives the ICS Manager's identifier.

application interface library (AIL)
A library that implicitly appears at the top of the library stack and typically exports the application interface to accessories. The INFOConnect Accessory AIL (IcAAP16.DLL) exports all session related interfaces of the INFOConnect Accessories API. Other AAPI functions are exported directly by the ICS Manager and are also available to INFOConnect accessories.

Application Type
See Open ID.
Glossary

B

branded component numbers
A supplier-specific identifier that uniquely identifies a component. See component number.

C

channel data
Global channel-related data for ICS libraries that is reusable on a per-session basis. Default data may be supplied by the library during template installation. Channel data may be configured by the user through channel configuration and associated with path data during path configuration. This is the data passed into IcLibOpenChannel.

CodeDir
The name of the directory that contains INFOConnect code files. CodeDir refers to either the [INFOConnect] CodeDir entry from WIN.INI or, if that does not exist, the directory from which the ICS Manager DLLs are executing.

Communications Manager
The ICS Manager component that provides the interface between the accessory and the library components. The Communications Manager handles loading the necessary libraries at session establishment.

communication path
See path.

communication session
See session.

component number
Identifiers used by the INFOConnect Connectivity Services configuration accessory to uniquely identify components. Component numbers are defined by the IC_COMPONENT data type. See Appendix A for more information.

configuration session
An instance of active configuration of a particular INFOConnect element (such as an INFOConnect path or library).

configuration accessory or configurator
An INFOConnect accessory that provides the user interface to the configuration functions for the INFOConnect Connectivity Services product. There may be more than one configuration accessory executing. The configuration accessory provided on the ICS runtime diskettes is referred to as the INFOConnect Manager.
Configuration Manager

The ICS Manager component that provides the configuration feature of INFOConnect by allowing access to the Database Manager. It also manages the interface between libraries during configuration.

context

A dynamically assigned identification for INFOConnect Connectivity Services loaded DLLs and registered accessories. It can be used to uniquely identify the accessory/library where the status and error messages are defined. A context is part of an IC_RESULT value.

class context string

Unique identification string that is used to obtain a unique context for loaded ICS components. The context string is defined in the component’s .HIC include file.

cooperative system

A system consisting of multiple components that may be executing on a single computer system or on different computer systems. INFOConnect Connectivity Services provides the communications layer between different components of the cooperative system when one of the components is running on a workstation GUI platform.
Glossary

D

data dictionary table
A table of IC_DICT_FIELDs followed by a single NULL value. This is a user-defined resource type with the type ID given as the DictRcType field value in the IC_DICT_NODE resource. The name IDs are computed using the TableFirst and TableCount field values of the same resource. Each table defines some portion of a library's configuration data.

Database Manager
The ICS Manager component that maintains the configuration database.

DataDir
The name of the directory that contains INFOConnect data files. DataDir refers to either the [INFOConnect] DataDir entry from WIN.INI or, if that does not exist, the Windows Directory.

DosLink
Client/Server-type DOS applications that run in Windows enhanced mode and utilize the ICS API for data communications.

DosLink API
A subset of the Accessory API that defines those INFOConnect data communications services available to DOS applications.

E

EIL
See external interface library.

exit-hook library
A special library that gains control from the ICS Installation Accessory at certain points during installation and deinstallation of the product. There may be only one exit-hook library per package. If this library exists, its filename is recorded in the package INF installation file.

external interface library (EIL)
A library that acts as an adaptor to a particular type of communications hardware or software. Each path is configured with a single EIL. EILs act as the point where a path connects to another "environmental context". This is often an external communications driver, but an EIL can also connect to another INFOConnect path and initiate another pass through the INFOConnect architecture. Applications, as well as other libraries in the path, are unaware of EILs.
G

generic component number
Identifies a component according to its function. See component number.

Graphical User Interface (GUI)
User presentation that consists of managing multiple objects on a single screen. The interface consists of windows, dialogs, keyboard, and, mouse support which together provide a high-level of consistency to the users perception of the system.

GUI
See Graphical User Interface.

H

.HIC include file
An include file provided by an ICS library component that contains the library's context string, library-specific statuses and errors, and definitions for each of the library's configuration tables along with field definitions for each field of the tables.

hidden path
A path configured as hidden will not appear for selection at runtime when an INFOConnect application or accessory opens a session without a pre-specified path. This is useful when a path is pre-configured for use by a particular application or accessory and so should not be chosen for use with other applications.

hidden template
A template configured as hidden will not appear for selection during path configuration. This is useful for administrators to configure templates that are not normally visible to the user.

hook library
A special purpose library that provides special features to the ICS Manager. See trace library, exit-hook library.
ICS Manager or The Manager

The backplane of the INFOConnect Connectivity Services product. The ICS Manager consists of a set of dynamic link libraries that control data communications as well as provide access to the INFOConnect API. The following components provide all of the features of the ICS product: Communication Manager, Configuration Manager, Database Manager, INFOConnect Manager, Installation Manager, and Utility Manager.

ICS path

See path.

IC_RESULT

An IC_RESULT is a small packet of data used to describe errors and statuses. Most INFOConnect functions and events return an IC_RESULT indicating success or failure. Functions exist to translate 'error' IC_RESULTS into displayable text strings. IC_RESULT consists of three parts: a context, a type, and a value. Utilities exist to extract the various parts from an IC_RESULT and to create an IC_RESULT from its parts.

ID (accessory ID/library ID)

An ID, or key, that identifies the fully qualified runfile name. It must be less than IC_MAXIDSIZE large, and is usually installed for the user during product installation.

IIL

See interprocess interface library.

.INF file

An installation script file for an INFOConnect package.

INFOCONN.CFG

This file contains configuration information for INFOConnect Connectivity Services and its currently configured paths.

INFOConnect library

Application Interface Libraries, Service Libraries, External Interface Libraries, Quick Configuration Libraries, and Hook Libraries are collectively referred to as INFOConnect libraries. Also see interprocess interface library and stack library.
INFOConnect Manager
The *ICS Manager* component that is the user interface to ICS. It provides both configurator and shell accessory features.

**installation accessory**
An INFOConnect utility that provides the user interface for installation, deinstallation, and quick configuration of INFOConnect components. The installation accessory provided on the ICS runtime diskettes is referred to as the Installation Manager. Also see local installation, standalone installation, subscribe installation, and publish installation.

**Installation Manager**
The *ICS Manager* component that provides the installation, deinstallation, and quick configuration features. Also see installation accessory.

**INSTMGR.CFG**
This file contains package information for INFOConnect Connectivity Services and its currently installed packages.

**interprocess interface library (IIL)**
A library that acts as both an AIL and an EIL. An IIL associates two sessions in different processes by internally linking the EIL role of one session to the AIL role of the other session. Libraries of this type are typically not included in path templates. The IIL is automatically included in sessions when an AIL requests a path that must be opened in a different process.

**L**

**library**
See INFOConnect library, multiplexing library.

**library channel**
See channel data.

**library ID**
See ID.

**library stack**
A stack of ICS libraries consisting of an application interface library, zero to 14 service libraries, and terminated by a single external interface library.
local installation
Installation (/L option) of a package on a workstation or server that redirects files destined for the Windows and Windows system directories to the installation destination directory. This option, therefore, affects the destination of files during standalone installation, publish installation, and subscribe installation.

M
Manager
See ICS Manager.

multiplexing library
A stack interface library that can support multiple communication sessions (where it is configured as an EIL) over another session. The sessions are associated with a channel in the EIL role which is associated with a lower level path. Typically, this lower level path is specified during channel configuration of the EIL role.

O
Open ID
The Open ID, also referred to as Application Type, is a library or accessory identifier that is used to narrow the list of available paths or templates during a selection.

P
path (ICS path or communication path)
Defines the hardware and software components (and their configurations) necessary for communicating between components of a cooperative application. It involves zero or more (up to 14) service libraries and one external interface library, along with their respective configurations. The path may involve communications within the system or to another computer. It is identified by a path ID.

path data
Data that is specific to ICS libraries and is unique on a per-session basis. Path data will be configured by the user through path configuration. This is the data passed into IcLibOpenSession.

path ID
A unique, user-assigned string of fifteen characters or less, containing no spaces, colons(:), or tildes(~) that identifies an ICS path.

path template
See template.
publish installation
Installation (/A option) of a package to a shared directory on a network by the network administrator. All necessary files are copied to various directories on the server, including the Windows and the Windows system directories. The package is shared by network users through a subscribe installation. Also see local installation.

Q
quick configuration library
A special purpose library that performs quick configuration for a library or a set of libraries in a package.

S
service library (SL)
A library that acts as a filter on the data and status messages which flow between an application interface library and external interface library. Zero or more service libraries can be stacked in a single INFOConnect path. Service libraries generally operate independently and are unaware of the other libraries in the path.

session (communication session)
An open or active instance of an ICS path. It has an associated session handle that is a unique integer used by INFOConnect Connectivity Services to identify the communication session.

session identification string
A string consisting of the path ID and, if multiple copies of the path can be active, a semicolon and the unique library-defined session ID.

session manager stack library
A stack interface library that can support multiple communication sessions (where it is configured as an EIL) over another session. The sessions are grouped into a session group (sometimes by using a channel). One or more alternate lower level paths may be configured for fallback when the primary lower level path is unavailable. This type of library filters the data stream for commands that reroute the session data to different applications.

shell or shell accessory
An INFOConnect utility that acts as the EXE portion of the ICS Manager. It must call the IcInitShell procedure before entering its message loop. Only one INFOConnect Shell can be running at any given time, and it may or may not include a configuration accessory.
Glossary

SL
See service library.

stack
See library stack.

stack interface library
A library that acts as both an AIL and an EIL. A stack interface library provides multiplexing or switching functions on lower level sessions. These types of libraries associate two sessions in the same process by internally linking the EIL role of one session to the AIL role of the other session. Libraries of this type can be included in path templates as an EIL (for use by higher level paths). Also see multiplexing library, switching library, session manager stack library.

stack library
See stack interface library.

standalone installation
Installation of a package on a workstation. All necessary files are copied to various directories on the workstation, including the Windows and the Windows system directories. Also see local installation.

subscribe installation
Installation (/N option) of a package from a network to a workstation that allows the workstation to access the shared copy of the package. Various files may be copied to the Windows and the Windows system directories. Also see local installation.

switching stack library
A stack interface library that stacks one session (where it is configured as an EIL) on top of another session and filters the data stream for commands to close and open the lower session.

system path
A path marked as system implies that the path’s associated Open ID is intended to reference a library rather than an accessory. This is convenient for low-level paths that are used as transport layers by higher level paths (usually by supplying the path ID as channel data for the EIL associated with the higher-level path template). Paths marked as system do not appear for selection during path configuration. These paths are normally configured automatically by the library component that uses these path. System paths should also be marked as hidden to prevent them from appearing for user selection at runtime.
**system template**

A template marked as *system* implies that the template's associated *Open ID* is intended to reference a library rather than an accessory. This is convenient for low-level paths that are used as transport layers by higher level paths (usually by supplying the path ID as *channel data* for the *EIL* associated with the higher-level template). System templates are typically used by library configuration and quick configuration libraries when creating system paths. Therefore, they are normally configured automatically by the library components that use these templates. Templates marked as system should also be marked as *hidden* in order to prevent them from appearing for user selection.

**template**

A stack of ICS libraries consisting of zero or more service libraries terminated by a single external interface library. The EIL may be associated with channel data. Templates are usually installed for the user during library installation and are selected during path configuration to create paths. Templates generally categorize the basic types of connections available on a workstation. This simplifies the path configuration process by reducing a large number of libraries to a small set of path templates.

**template ID**

A string, or key, less than IC_MAXIDSIZE large that identifies the *template*.

**trace library**

A special service library that traces session communication so that session activity can be monitored.

**trace log library**

A special library that manages a log file by writing information to it.

**Utility Manager**

The *ICS Manager* component that provides internal utilities.
Glossary

X

XVT

XVT is a software toolkit produced by XVT Software Inc. that provides graphical presentation services like windows, list boxes, scroll bars, etc. to applications. Developers using XVT instead of directly using the underlying window system (i.e. making direct calls to Windows functions) may readily port their applications to any of the GUIs that Unisys offers on its workstations. INFOConnect applications are strongly encouraged, but not required, to use the XVT toolkit instead of the native presentation services.
Index

A

accessory
  definition, 6-1
  standard IDs (keys), A-1
  statuses from, B-2, B-10
  statuses to, B-7, B-10, B-12
Windows API, 2-2
XVT/Win API, 2-3
accessory API, (See also memory management API and general utilities)
  ic_change_handle, 3-141
  ic_close_session, 3-142
  ic_default_error_proc, 3-143
  ic_exit_ok, 3-146
  ic_get_path_id, 3-153
  ic_get_session_id, 3-155
  ic_get_session_info, 3-156
  ic_get_string, 3-157
  ic_init_ics, 3-162
  ic_lcl, 3-163
  ic_open_accessory, 3-164
  ic_open_session, 3-167
  ic_rcv, 3-170
  ic_register_msg_session, 3-172
  ic_set_error, 3-176
  ic_set_status, 3-177
  ic_xmt, 3-178
IcChangeHandle, 3-4
IcCloseSession, 3-8
IcDefaultErrorProc, 3-12
IcExitOk, 3-21
IcGetPathID, 3-35
IcGetSessionID, 3-43
IcGetSessionInfo, 3-44
IcGetString, 3-45
IcInitIcs, 3-47
IcLcl, 3-49
IcOpenAccessory, 3-97
IcOpenSession, 3-100
IcRcv, 3-104
IcRegisterMsgSession, 3-113
IcSelectPath, 3-121
IcSetError, 3-123
IcSetStatus, 3-127
IcXmt, 3-133
accessory utilities
  IcGetCmdlineOption, 3-25
AIL, (See application interface library)
application interface library, 5-8

B

basic session management API
  ic_close_session, 3-142
  ic_exit_ok, 3-146
  ic_init_ics, 3-162
  ic_lcl, 3-163
  ic_open_accessory, 3-164
  ic_open_session, 3-167
  ic_rcv, 3-170
  ic_register_msg_session, 3-172
  ic_xmt, 3-178
IcCloseSession, 3-8
IcExitOk, 3-21
IcInitIcs, 3-47
IcLcl, 3-49
IcOpenAccessory, 3-97
IcOpenSession, 3-100
IcRcv, 3-104
IcRegisterMsgSession, 3-113
IcXmt, 3-133
blocking, 5-54
branded component numbers, 5-7, A-4
buffers, (See memory management API)

C

CHANNELID, 5-1
Client/Server Applications, (See DosLink)
component numbers, A-4
configuration API
Index

IcNotifyConfig, 3-95
IcSelectPath, 3-121

D
DosLink
displaying error strings, 5-17
DosLink API, 2-5
DosLink Specific API
IcCreateHandle, 3-9
IcCreateHwnd, 3-10
IcCreateSession, 3-11
IcDestroyHandle, 3-17
IcDestroyHwnd, 3-18
IcDestroySession, 3-19
IcgetNextEvent, 3-34
IcGetServiceName, 3-42
IcHandleOffset, 3-46
IcNextEvent, 3-93
IcRegisterCallback, 3-111
IcSetServerInfo, 3-124
DosLink-specific statuses, B-16
DOSLINK_SINFO, B-16

E
E_IC_ERROR, 4-3
E_IC_LCL_RESULT, 3-163, 3-179, 4-4
E_IC_NEWPATH, 3-152, 4-5
E_IC_NULLEVENT, 3-172, 4-6
E_IC_RCV_DONE, 3-170, 4-7
E_IC_RCV_ERROR, 3-170, 4-8
E_IC_SESSION_CLOSE, 3-142, 4-9
E_IC_SESSION_EST, 3-142, 3-168, 4-10
E_IC_STATUS, 4-11
E_IC_STATUS_RESULT, 3-177, 4-12
E_IC_XMT_DONE, 3-179, 4-13
E_IC_XMT_ERROR, 3-179, 4-14
error handling API
ic_default_error_proc, 3-143
ic_get_string, 3-157
ic_set_error, 3-176
icDefaultErrorProc, 3-12
icGetString, 3-45
icSetError, 3-123
errors
IcACOMS, C-49
IcHLCNTS, C-52
IcLCW, C-56
IcLocal, C-57
IcMon, C-58
IcNBIOS, C-60
IcTCP, C-65
IcTELNET, C-66
IcTrace, C-68
IcTTY, C-69
IcXNS, C-75
standard ICS, C-2
EVENT, 5-2
extended status, 5-55
external interface library, 5-8
standard IDs (keys), A-3

F
field
flags, 5-20
standard, 5-81, 5-82, 5-83
types, 5-22
flags
field, 5-20
library, 5-27
session, 5-49
table, 5-66

G
general utilities
IC_CHECK_DATAFLAGS, 3-6
IC_CHECK_RESULT_SEVERE, 3-7
ic_deregister_accessory, 3-145
ic_get_context, 3-148
ic_get_context_string, 3-149
ic_get_infoconnect_dir, 3-150
ic_get_new_path, 3-151
ic_get_path_names, 3-154
IC_GET_RESULT_CONTEXT, 3-37
IC_GET_RESULT_SUBTYPE, 3-38
IC_GET_RESULT_SUBVALUE, 3-39
IC_GET_RESULT_TYPE, 3-40
IC_GET_RESULT_VALUE, 3-41
ic_make_result, 3-80
ic_register_accessory, 3-171
ic_run_accessory, 3-174
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IcDeregisterAccessory</td>
<td>3-16</td>
</tr>
<tr>
<td>IcGetContext</td>
<td>3-27</td>
</tr>
<tr>
<td>IcGetContextString</td>
<td>3-29</td>
</tr>
<tr>
<td>IcGetINFOConnectDir</td>
<td>3-30</td>
</tr>
<tr>
<td>IcGetNewPath</td>
<td>3-32</td>
</tr>
<tr>
<td>IcGetPathNames</td>
<td>3-36</td>
</tr>
<tr>
<td>IcMgrTraceBuffer</td>
<td>3-88</td>
</tr>
<tr>
<td>IcMgrTraceResult</td>
<td>3-90</td>
</tr>
<tr>
<td>IcReadBuffer</td>
<td>3-106</td>
</tr>
<tr>
<td>IcRegisterAccessory</td>
<td>3-110</td>
</tr>
<tr>
<td>IcRunAccessory</td>
<td>3-116</td>
</tr>
<tr>
<td>IcRunHelp3</td>
<td>3-118</td>
</tr>
<tr>
<td>IcWriteBuffer</td>
<td>3-129</td>
</tr>
<tr>
<td>ic_buf_free</td>
<td>3-137</td>
</tr>
<tr>
<td>ic_buf_lock</td>
<td>3-138</td>
</tr>
<tr>
<td>ic_buf_realloc</td>
<td>3-139</td>
</tr>
<tr>
<td>ic_buf_unlock</td>
<td>3-140</td>
</tr>
<tr>
<td>IC_BUFHND</td>
<td>5-5</td>
</tr>
<tr>
<td>IC_BUILD_REVISION</td>
<td>5-5</td>
</tr>
<tr>
<td>IC_CALLBACK</td>
<td>5-5</td>
</tr>
<tr>
<td>ic_change_handle</td>
<td>3-141</td>
</tr>
<tr>
<td>IC_CHECK_DATAFLAGS</td>
<td>3-6</td>
</tr>
<tr>
<td>IC_CHECK_RESULT_SEVERE</td>
<td>3-7</td>
</tr>
<tr>
<td>ic_close_session</td>
<td>3-142</td>
</tr>
<tr>
<td>IC_CMD_...</td>
<td>(See IC_COMMAND)</td>
</tr>
<tr>
<td>IC_CODEDIR</td>
<td>5-16</td>
</tr>
<tr>
<td>IC_COMMAND</td>
<td>3-73, 5-6</td>
</tr>
<tr>
<td>IC_COMMGR_INITIALIZED</td>
<td>4-11, 4-26</td>
</tr>
<tr>
<td>IC_COMMGR_TERMINATED</td>
<td>4-11, 4-26</td>
</tr>
<tr>
<td>IC_COMPONENT</td>
<td>5-7, A-4</td>
</tr>
<tr>
<td>IC_COMPONENT_TYPE</td>
<td>5-7</td>
</tr>
<tr>
<td>IC_CONNECT_STATUS</td>
<td>5-49</td>
</tr>
<tr>
<td>IC_DATADIR</td>
<td>5-16</td>
</tr>
<tr>
<td>IC_DATAFLAGS</td>
<td>5-60</td>
</tr>
<tr>
<td>IC_DEBUG</td>
<td>3-48, 5-10</td>
</tr>
<tr>
<td>ic_default_errorProc</td>
<td>3-143</td>
</tr>
<tr>
<td>IC_DELETE_CONFIG</td>
<td>5-36</td>
</tr>
<tr>
<td>ic_deregister_accessory</td>
<td>3-145</td>
</tr>
<tr>
<td>IC_DICT_FIELD</td>
<td>5-12, 5-44</td>
</tr>
<tr>
<td>IC_DICT_NODE</td>
<td>5-13</td>
</tr>
<tr>
<td>IC_DICT_TABLE</td>
<td>5-14, 5-15</td>
</tr>
<tr>
<td>IC_DIRECTORYTYPES</td>
<td>3-30, 5-16</td>
</tr>
<tr>
<td>IC_EMU_LEVEL</td>
<td>5-17</td>
</tr>
<tr>
<td>IC_ERROR</td>
<td>4-15</td>
</tr>
<tr>
<td>IC_ERROR_INFO</td>
<td>5-17</td>
</tr>
<tr>
<td>IC_ERROR_MASK</td>
<td>5-18</td>
</tr>
<tr>
<td>IC_ERROR_SEVERE</td>
<td>5-18</td>
</tr>
<tr>
<td>IC_ERROR_TERMINATE</td>
<td>5-19</td>
</tr>
<tr>
<td>IC_ERROR_WARNING</td>
<td>5-19</td>
</tr>
<tr>
<td>ic_exit_ok</td>
<td>3-146</td>
</tr>
<tr>
<td>IC_FF_...</td>
<td>(See IC_FIELD_FLAGS)</td>
</tr>
<tr>
<td>IC_FIELDFLAGS</td>
<td>5-20</td>
</tr>
<tr>
<td>IC_FIELDTYPE</td>
<td>5-22</td>
</tr>
<tr>
<td>IC_FIELDTYPE [IC_FT_UNSIGNED]</td>
<td>5-30</td>
</tr>
<tr>
<td>IC_ACCESSORY</td>
<td>5-7</td>
</tr>
<tr>
<td>IC_ADD_CONFIG</td>
<td>5-35</td>
</tr>
<tr>
<td>IC_APILIBRARY</td>
<td>5-7</td>
</tr>
<tr>
<td>IC_APPINTERFACE</td>
<td>5-8</td>
</tr>
<tr>
<td>IC_APIPLIBRARY</td>
<td>5-8</td>
</tr>
<tr>
<td>IC_BASEREVISION</td>
<td>5-4</td>
</tr>
<tr>
<td>IC_BASEVERSION</td>
<td>5-4</td>
</tr>
<tr>
<td>ic_buf_alloc</td>
<td>3-136</td>
</tr>
</tbody>
</table>

**H**

<table>
<thead>
<tr>
<th>generic component numbers</th>
<th>5-7, A-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>global buffers</td>
<td>(See memory management API)</td>
</tr>
</tbody>
</table>
Index

ic_get_infoconnect_dir, 3-150
ic_get_new_path, 3-151, 4-5
ic_get_path_id, 3-153
ic_get_path_ids, 3-154
IC_GET_RESULT_CONTEXT, 3-37
IC_GET_RESULT_SUBTYPE, 3-38
IC_GET_RESULT_SUBVALUE, 3-39
IC_GET_RESULT_TYPE, 3-40
IC_GET_RESULT_VALUE, 3-41
ic_get_session_id, 3-155
ic_get_session_info, 3-156
ic_get_string, 3-157
ic_free, 3-158
ic_glock, 3-159
ic_grealloc, 3-160
ic_gunlock, 3-161
IC_HEADER_3_0, 5-25
IC_HEADER_SIZE, 5-25
IC_HOOKLIBRARY, 5-8
IC_INTERFACE, 5-8
IC_IPCINTERFACE, 5-8
IC_KEY_SERIALNUM, 5-26
IC_LASTEVENT, 4-16
ic_lcl, 3-163, 4-4
IC_LCL_FLAGS, 3-5, 3-60, 5-26
IC_LCL_RCVXMT, 5-26
IC_LCLRESULT, 3-49, 3-134, 4-17
IC_LF_..., (See IC_LIBRARY_FLAGS)
IC_LIBRARY, 5-8
IC_LIBRARY_FLAGS, 5-27
IC_MAKE_RESULT, 3-80
IC_MANAGER, 5-9
IC_MASTERDIR, 5-16
IC_MAXACCESSORYIDLEN, 5-27
IC_MAXACCESSORYIDSIZE, 5-28
IC_MAXCHANNELIDLEN, 5-28
IC_MAXCHANNELIDSIZE, 5-28
IC_MAXCONNECTEDPATHIDLEN, 5-28
IC_MAXDESCRIPTIONSIZE, 5-29
IC_MAXERRORINSERT, 5-29
IC_MAXERRORSTRING, 5-29
IC_MAXFILENAMESIZE, 5-30
IC_MAXFILENAMESIZE, 5-30
IC_MAXLIBRARYIDLEN, 5-30
IC_MAXLIBRARYIDSIZE, 5-31
IC_MAXPACKAGENAMELEN, 5-31
IC_MAXPATHIDLEN, 5-31
IC_MAXPATHIDSIZE, 5-31
IC_MAXSTRINGLENGTH, 5-33
IC_MAXTEMPLATEIDLEN, 5-33
IC_MAXTEMPLATEIDSIZE, 5-33
IC_MAXVENDORNAMELEN, 5-34
IC_MAXVENDORNAMESIZE, 5-34
IC_MAXWSIDSIZE, 5-34
IC_MEMHND, 5-34
IC_MGR_INIT, 5-16
IC_MINOR_VERSION, 5-35
IC_MSG_CONFIG, 3-96, 5-35
IC_NEWPATH, 3-32, 4-18
IC_NEXTEVENT_FLAGS, 5-37
IC_NULLEVENT, 4-19
IC_OK, 5-38
ic_open_accessory, 3-164
IC_OPEN_OPTIONS, 3-62, 3-66, 5-38
ic_open_session, 3-167
IC_PACKAGE, 5-39
IC_PATH_FLAGS, 5-39
IC_PRINT_SUMMARY, 5-39
IC_QUICKCONFIG, 5-9
IC_RC_NODE, 4-23, 4-24, 4-26, 5-40, 5-49
ic_rcv, 3-170
IC_RCVDONE, 3-46, 3-105, 4-20, 5-60
IC_RECEIVED, 3-105, 4-21
IC_RECORD_INFO, 5-43
IC_RECORD_SIZE, 5-44
IC_REFRESH_CONFIG, 5-36
ic_register_accessory, 3-171
ic_register_msg_session, 3-172
IC_RESULT, 5-44
IC_RESULT_CONTEXT, 5-44
IC_RESULT_CONTEXT_INVALID, 5-46
IC_RESULT_SUBTYPE, 5-46
IC_RESULT_SUBVALUE, 5-47
IC_RESULT_TYPE, 5-44
IC_RESULT_VALUE, 5-44
ic_run_accessory, 3-174
IC_SENDSTATUS, 4-22
IC_SERIALNUM, 5-48
IC_SERVICE, 5-9
IC_SESSION_FLAGS, 5-49
IC_SESSIONCLOSED, 3-8, 3-52, 3-61, 3-83, 4-23
IC_SESSIONESTABLISHED, 3-8, 3-52, 3-102, 4-24
ic_set_error, 3-176
ic_set_status, 3-177, 4-12
IC_SF_... (See IC_SESSION_FLAGS)
IC_SINFO, 3-44, 3-54, 3-82, 5-50
IC_STACKINTERFACE, 5-9
IC_STATUS, 4-21, 4-26
IC_STATUS_BLOCKING, 5-50, 5-54, B-2
IC_STATUS_BUFFER, 5-55, B-2
IC_STATUS_COMMGR, 3-21, 3-53, 4-22, 5-56, B-12
IC_STATUS_CONNECT, 3-124, 5-50, 5-57, B-3, B-7
IC_STATUS_CONTROL, 5-59, B-8, B-11
IC_STATUS_DATAFLAGS, 5-60, B-4, B-10
IC_STATUS_FKEY, 5-62, B-5
IC_STATUS_LINESTATE, 5-63, B-8
IC_STATUS_REACTIVATE, 5-50, 5-64, B-6
IC_STATUS_TRANS, 5-65, B-6
IC_STATUS_UTS, 5-66, B-14
IC_STATUSBUF, 5-52
IC_STATUSRESULT, 3-127, 4-27, 5-53, 5-55
IC_TABLE_FLAGS, 5-66
IC_TABLETYPE, 5-68
IC_TemplateBegin, 5-69
IC_TemplateChannel, 5-69
IC_TemplateConfig, 5-70
IC_TemplateConfigTable, 5-71
IC_TemplateDescription, 5-71
IC_TemplateEnd, 5-72
IC_TemplateFlags, 5-72
IC_TemplateInit, 5-72
IC_TemplateLibrary, 5-74
IC_TemplateOpenID, 5-74
IC_TemplateTerm, 5-75
IC_TF_... (See IC_TABLE_FLAGS)
IC_TIMER, 3-93, 4-28
IC_UPDATE_CONFIG, 5-36
IC_UPGRADE_INFO, 5-75
IC_VER, 5-76
IC_VER_INFO, 5-77
IC_VER_UPGRADE, 5-75
IC_VERIFY, 3-76, 5-78
IC_VERIFY_OK, 5-79
IC_VERSION_..., 5-80
IC_VERSION_FILE, 5-79
IC_VERSION_PRODUCT, 5-80
IC_VERSION_STRING, 5-5, 5-80
ic_xmt, 3-178
IC_XMTDONE, 3-46, 3-134, 4-29
IC_XMTERROUR, 3-134, 4-30
IcACOMS errors, C-49
IcAllocBuffer, 3-3
IcChangeHandle, 3-4
IcCloseSession, 3-8
IcCreateHandle, 3-9
IcCreateHwnd, 3-10
IcCreateSession, 3-11, 3-101
IcDeleteLibraryConfig, 3-14
IcDeregisterAccessory, 3-16
IcDestroyHandle, 3-17
IcDestroyHwnd, 3-18
IcDestroySession, 3-19
IcDialogConfig, 3-20, 3-73
IcExitOk, 3-21, 5-56
IcFreeBuffer, 3-22
IcGetBufferSize, 3-23
IcGetChannelID, 3-24
IcGetCmdlineOption, 3-25
IcGetContext, 3-27
IcGetContextID, 3-28, 3-115
IcGetContextString, 3-29
IcGetINFOConnectDir, 3-30
IcGetLibraryDefault, 3-31
IcGetNewPath, 3-32, 4-18
IcGetNextEvent, 3-33, 3-34, 5-37
IcGetPathID, 3-35
IcGetPathNames, 3-36
IcGetServiceName, 3-42
IcGetSessionID, 3-43
IcGetSessionInfo, 3-44
IcGetString, 3-45
IcHandleOffset, 3-46
IcHLCNTS errors, C-52
IcInitIcs, 3-47
IcIsDebug, 3-48
IcLcl, 3-49, 4-17
IcLCW errors, C-56
IcLibCloseChannel, 3-50
Index

IcLibCloseSession, 3-51
IcLibEvent, 3-52
IcLibGetSessionInfo, 3-54
IcLibGetString, 3-55
IcLibIdentifySession, 3-57, 5-33
IcLibInstall, 3-58
IcLibLcl, 3-60
IcLibOpenChannel, 3-62
IcLibOpenSession, 3-57, 3-65
IcLibPrintConfig, 3-68, 5-32
IcLibRecv, 3-70
IcLibSetResult, 3-71
IcLibTerminate, 3-72
IcLibUpdateConfig, 3-73, 5-6
IcLibVerifyConfig, 3-76, 5-75
IcLibXmt, 3-78
IcLocal errors, C-57
IcLockBuffer, 3-79
IcMgrEilEvent, 3-81
IcMgrGetSessionInfo, 3-82
IcMgrLcl, 3-60, 3-83, 4-24
IcMgrRecv, 3-84, 4-24
IcMgrSendEvent, 3-52, 3-85, 4-24
IcMgrSetResult, 3-87, 4-24
IcMgrTraceBuffer, 3-88
IcMgrTraceResult, 3-90
IcMgrXmt, 3-92, 4-24
IcMon errors, C-58
IcNBIOS errors, C-60
IcNextEvent, 3-93, 3-111, 5-37
IcNotifyConfig, 3-95
IcOpenAccessory, 3-97
IcOpenSession, 3-100, 3-114
IcRecv, 3-104
IcReadBuffer, 3-106
IcReadLibraryConfig, 3-107
IcReAllocBuffer, 3-109
IcRegisterAccessory, 3-110, 3-127
IcRegisterCallback, 3-11, 3-19, 3-111
IcRegisterMsgSession, 3-100, 3-113, 4-1, 4-16
IcReleaseContextID, 3-2, 3-28, 3-115
IcRunAccessory, 3-99, 3-116
IcRunHelp3, 3-118
IcRunLibHelp, 3-120
ICS errors, C-2
IcSelectPath, 3-121
IcSetError, 3-123
IcSetServerInfo, 3-11, 3-124
IcSetSessionError, 3-55, 3-125
IcSetStatus, 3-127, 4-27
ICSTD_ACTIVECHANNEL, 5-81
ICSTD_ACTIVEPATH, 5-81
ICSTD_ACTIVEPATHCHANNEL, 5-82
ICSTD_CHANNEL, 5-82
ICSTD_PATH, 5-82
ICSTD_PATHCHANNEL, 5-83
IcTCP errors, C-65
IcTELNET errors, C-66
IcTrace errors, C-68
IcTTY errors, C-69
IcUnlockBuffer, 3-128
IcWriteBuffer, 3-129
IcWriteLibraryConfig, 3-131
IcXmt, 3-133
IcXNS errors, C-75
ICXVT_CONFIG, 5-83
ICXVTWIN, 5-83
INFOConnect Development Kit, 1-1
interprocess interface library, 5-8

K

keys
IC_FF_ALTERNATE_KEY, 5-20
IC_FF_LINK_KEY, 5-20
IC_FF_LINK_KEY_CHANNEL, 5-20
IC_FF_LINK_KEY_PATH, 5-21
IC_FF_PRIMARY_KEY, 5-21
referencing, 5-26
standard accessory, A-1
standard external interface library, A-3
standard service library, A-2
Index

L

library
application interface, 5-8
external interface, 5-8
hook, 5-8
interprocess interface, 5-8
quick configuration, 5-9
service, 5-9
stacking interface, 5-9
statuses from, B-7
statuses to, B-2
library API
entry points provided by, 2-10
utilities for development of, 2-12
library entry points, 2-10
IcLibCloseChannel, 3-50
IcLibCloseSession, 3-51
IcLibEvent, 3-52
IcLibGetSessionInfo, 3-54
IcLibGetString, 3-55
IcLibIdentifySession, 3-57
IcLibInstall, 3-58
IcLibLcl, 3-60
IcLibOpenChannel, 3-62
IcLibOpenSession, 3-65
IcLibPrintConfig, 3-68
IcLibRev, 3-70
IcLibSetResult, 3-71
IcLibTerminate, 3-72
IcLibUpdateConfig, 3-73
IcLibVerifyConfig, 3-76
IcLibXmt, 3-78
library utilities, (See also general utilities)
IcAddRefContextID, 3-2
IcDeleteLibraryConfig, 3-14
IcDialogConfig, 3-20
IcGetChannelID, 3-24
IcGetConfigID, 3-28
IcGetLibraryDefault, 3-31
IcIsDebug, 3-48
IcMgrEilEvent, 3-81
IcMgrGetSessionInfo, 3-82
IcMgrLcl, 3-83
IcMgrRev, 3-84
IcMgrSendEvent, 3-85
IcMgrSetResult, 3-87
IcMgrXmt, 3-92
IcNotifyConfig, 3-95
IcReadLibraryConfig, 3-107
IcReleaseContextID, 3-115
IcRunLibHelp, 3-120
IcSetSessionError, 3-125
IcWriteLibraryConfig, 3-131
LPHIC_CHANNEL, 5-84
LPHIC_SESSION, 5-84
LPIC_RESULT_CONTEXT, 5-84
LPIC_SINFO, 5-84
LPIC_STATUSBUF, 5-85
LPIC_UPGRADE_INFO, 5-85
LPIC_VER_INFO, 5-85

M

memory management API
ic_buf_alloc, 3-136
ic_buf_free, 3-137
ic_buf_lock, 3-138
ic_buf_realloc, 3-139
ic_buf_unlock, 3-140
ic_galloc, 3-147
ic_gfree, 3-158
ic_glock, 3-159
ic_grealloc, 3-160
ic_gunlock, 3-161
IcAllocBuffer, 3-3
IcFreeBuffer, 3-22
IcGetBufferSize, 3-23
IcLockBuffer, 3-79
IcReAllocBuffer, 3-109
IcUnlockBuffer, 3-128
Windows, 2-6
XVT/Win, 2-6

N

NOREF, 3-135
NULL_HIC_CHANNEL, 5-86
NULL_HIC_CONFIG, 5-86
NULL_HIC_SESSION, 5-86
NULL_HIC_STATUSBUF, 5-86
NULL_IC_BUFXND, 5-87
NULL_IC_MEMHND, 5-87
Index

O
OpenID,  5-74

P
PATHID,  5-87

Q
quick configuration library,  5-9

R
record
information,  5-43
results, API for
checking severity,  3-7
creating,  3-80
retrieving parts,  3-37, 3-38, 3-39, 3-40, 3-41

S
service library,  5-9
standard IDs (keys),  A-2
session information,  5-50
session management API
ic_change_handle,  3-141
ic_close_session,  3-142
ic_exit_ok,  3-146
ic_get_path_id,  3-153
ic_get_session_id,  3-155
ic_get_session_info,  3-156
ic_init_ics,  3-162
ic_lcl,  3-163
ic_open_accessory,  3-164
ic_open_session,  3-167
ic_rcv,  3-170
ic_register_msg_session,  3-172
ic_set_status,  3-177
ic_xmt,  3-178
IcChangeHandle,  3-4
IcCloseSession,  3-8
IcExitOk,  3-21
IcGetPathID,  3-35
IcGetSessionID,  3-43
IcGetSessionInfo,  3-44
IcInitIcs,  3-47
IcLcl,  3-49
IcOpenAccessory,  3-97
IcOpenSession,  3-100
IcRcv,  3-104
IcRegisterMsgSession,  3-113
IcSetStatus,  3-127
IcXmt,  3-133
standard configurator results
by value
severe
(Value 100),  C-36
(Value 101),  C-45
(Value 102),  C-45
(Value 103),  C-38
(Value 104),  C-39
(Value 105),  C-37
(Value 106),  C-37
(Value 107),  C-41
(Value 108),  C-39
(Value 109),  C-38
(Value 110),  C-41
(Value 111),  C-38
(Value 112),  C-40
(Value 113),  C-34
(Value 114),  C-40
(Value 115),  C-42
(Value 116),  C-40
(Value 117),  C-47
(Value 118),  C-47
(Value 119),  C-46
(Value 120),  C-47
(Value 121),  C-46
(Value 122),  C-46
(Value 123),  C-48
(Value 124),  C-48
(Value 125),  C-45
(Value 126),  C-42
(Value 127),  C-36
(Value 134),  C-34
(Value 135),  C-42
(Value 136),  C-43
(Value 137),  C-44
(Value 138),  C-44
(Value 139),  C-43
(Value 140),  C-35
(Value 143),  C-35
(Value 160),  C-37
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Value 161), C-39</td>
</tr>
<tr>
<td>(Value 162), C-41, C-44</td>
</tr>
<tr>
<td>warning (Value 128), C-43</td>
</tr>
<tr>
<td>(Value 129), C-47</td>
</tr>
<tr>
<td>(Value 130), C-42</td>
</tr>
<tr>
<td>(Value 131), C-36</td>
</tr>
<tr>
<td>(Value 132), C-36</td>
</tr>
<tr>
<td>(Value 133), C-35</td>
</tr>
<tr>
<td>(Value 141), C-34</td>
</tr>
<tr>
<td>(Value 142), C-46</td>
</tr>
<tr>
<td>standard context (Value 128), C-43</td>
</tr>
<tr>
<td>configuration accessory, 5-45 (Value 128), C-43</td>
</tr>
<tr>
<td>database utility, 5-45 (Value 128), C-43</td>
</tr>
<tr>
<td>ICS Manager, 5-46 (Value 128), C-43</td>
</tr>
<tr>
<td>utilities, 5-45 (Value 128), C-43</td>
</tr>
<tr>
<td>standard ICS results (Value 128), C-43</td>
</tr>
<tr>
<td>by value informational (Value 0), C-33</td>
</tr>
<tr>
<td>(Value 320), C-33</td>
</tr>
<tr>
<td>(Value 2000), C-10</td>
</tr>
<tr>
<td>(Value 2001), C-20</td>
</tr>
<tr>
<td>(Value 2002), C-29</td>
</tr>
<tr>
<td>(Value 2003), C-3</td>
</tr>
<tr>
<td>(Value 2004), C-3</td>
</tr>
<tr>
<td>(Value 2005), C-33</td>
</tr>
<tr>
<td>(Value 2006), C-32</td>
</tr>
<tr>
<td>(Value 2007), C-32</td>
</tr>
<tr>
<td>(Value 2008), C-17</td>
</tr>
<tr>
<td>(Value 2009), C-30</td>
</tr>
<tr>
<td>(Value 2010), C-30</td>
</tr>
<tr>
<td>(Value 2011), C-30</td>
</tr>
<tr>
<td>(Value 2012), C-29</td>
</tr>
<tr>
<td>(Value 2013), C-4</td>
</tr>
<tr>
<td>severe (Value 1), C-28</td>
</tr>
<tr>
<td>(Value 2), C-24</td>
</tr>
<tr>
<td>(Value 3), C-18</td>
</tr>
<tr>
<td>(Value 4), C-8</td>
</tr>
<tr>
<td>(Value 5), C-12</td>
</tr>
<tr>
<td>(Value 6), C-32</td>
</tr>
<tr>
<td>(Value 7), C-14</td>
</tr>
<tr>
<td>(Value 8), C-13</td>
</tr>
<tr>
<td>(Value 9), C-15</td>
</tr>
<tr>
<td>(Value 10), C-24</td>
</tr>
<tr>
<td>(Value 11), C-7</td>
</tr>
<tr>
<td>(Value 12), C-7</td>
</tr>
<tr>
<td>(Value 300), C-8</td>
</tr>
<tr>
<td>(Value 301), C-10</td>
</tr>
<tr>
<td>(Value 302), C-9</td>
</tr>
<tr>
<td>(Value 304), C-23</td>
</tr>
<tr>
<td>(Value 305), C-14</td>
</tr>
<tr>
<td>(Value 306), C-15</td>
</tr>
<tr>
<td>(Value 307), C-20</td>
</tr>
<tr>
<td>(Value 308), C-21</td>
</tr>
<tr>
<td>(Value 309), C-19</td>
</tr>
<tr>
<td>(Value 500), C-12</td>
</tr>
<tr>
<td>(Value 502), C-13</td>
</tr>
<tr>
<td>(Value 503), C-11</td>
</tr>
<tr>
<td>(Value 504), C-22</td>
</tr>
<tr>
<td>(Value 505), C-23</td>
</tr>
<tr>
<td>(Value 506), C-31</td>
</tr>
<tr>
<td>(Value 507), C-12</td>
</tr>
<tr>
<td>(Value 508), C-16</td>
</tr>
<tr>
<td>(Value 509), C-7</td>
</tr>
<tr>
<td>(Value 510), C-22</td>
</tr>
<tr>
<td>(Value 600), C-17</td>
</tr>
<tr>
<td>(Value 601), C-23</td>
</tr>
<tr>
<td>(Value 602), C-23</td>
</tr>
<tr>
<td>(Value 603), C-21</td>
</tr>
<tr>
<td>(Value 604), C-31</td>
</tr>
<tr>
<td>(Value 605), C-16</td>
</tr>
<tr>
<td>(Value 607), C-17</td>
</tr>
<tr>
<td>(Value 608), C-19</td>
</tr>
<tr>
<td>(Value 609), C-16</td>
</tr>
<tr>
<td>(Value 610), C-20</td>
</tr>
<tr>
<td>(Value 611), C-10</td>
</tr>
<tr>
<td>(Value 612), C-11</td>
</tr>
<tr>
<td>(Value 613), C-31</td>
</tr>
<tr>
<td>(Value 614), C-21</td>
</tr>
<tr>
<td>(Value 615), C-15</td>
</tr>
<tr>
<td>(Value 700), C-6</td>
</tr>
<tr>
<td>(Value 701), C-4</td>
</tr>
<tr>
<td>(Value 702), C-4</td>
</tr>
<tr>
<td>(Value 703), C-5</td>
</tr>
<tr>
<td>(Value 704), C-5</td>
</tr>
<tr>
<td>(Value 705), C-5</td>
</tr>
<tr>
<td>(Value 706), C-6</td>
</tr>
<tr>
<td>(Value 800), C-7</td>
</tr>
<tr>
<td>(Value 801), C-6</td>
</tr>
<tr>
<td>(Value 900), C-13</td>
</tr>
<tr>
<td>(Value 901), C-14</td>
</tr>
<tr>
<td>(Value 902), C-2</td>
</tr>
<tr>
<td>(Value 903), C-19</td>
</tr>
<tr>
<td>(Value 904), C-25</td>
</tr>
<tr>
<td>(Value 905), C-26</td>
</tr>
<tr>
<td>(Value 906), C-11</td>
</tr>
<tr>
<td>(Value 907), C-28</td>
</tr>
<tr>
<td>(Value 908), C-22</td>
</tr>
</tbody>
</table>
Index

T

table, configuration
flags, 5-66
IC_TABLETYPE, 5-68
template configuration, 5-69
transaction monitoring support, 5-65

U

utilities
for library development, 2-12
general, 2-7
Windows, 2-8
XVT/Win, 2-9
UTS-specific statuses, B-14

V

VER_FILEDESCRIPTION_STR, 5-88
VER_FILESUBTYPE, 5-89
VER_FILETYPE, 5-89
VER_INTERNALNAME_STR, 5-90

W

window state options, 3-99, 3-117, 3-165, 3-175
Windows API
accessory API, 2-2
DosLink API, 2-5
general utilities, 2-8
library API, 2-10
memory management API, 2-6
Windows messages
"IC_Error", 4-15
"IC_LclResult", 4-17
"IC_NewPath", 4-18
"IC_RcvDone", 4-20
"IC_RcvError", 4-21
"IC_SessionClosed", 4-23
"IC_SessionEstablished", 4-24
"IC_Status", 4-26
"IC_StatusResult", 4-27
"IC_Timer", 4-28
"IC_XmtDone", 4-29
"IC_XmtError", 4-30
IC_ERROR, 4-15
IC_LASTEVENT, 4-16
IC_LCLRESULT, 4-17
IC_NEWPAT, 4-18
IC_NULLEVENT, 4-19
IC_RXPDONE, 4-20
IC_RXPERROR, 4-21
IC_SENDSTATUS, 4-22
IC_SESSIONCLOSED, 4-23
IC_SESSIONESTABLISHED, 4-24
IC_STATUS, 4-26
IC_STATUSRESULT, 4-27
IC_TIMER, 4-28
IC_XMTDONE, 4-29
IC_XMTER, 4-30

DosLink-specific, B-16
from accessory to accessory, B-10
from accessory to library, B-2
from ICS to accessory, B-12
from library to accessory, B-7
UTS-specific, B-14
supplier-specific (branded) component numbers, A-4
Index

X

XVT
ICXVTCONFIG, 5-83
ICXVVTWIN, 5-83

XVT events
E_IC_ERROR, 4-3
E_IC_LCL_RESULT, 4-4
E_IC_NEWPATH, 4-5
E_IC_NULLEVENT, 4-6
E_IC_RCV_DONE, 4-7
E_IC_RCV_ERROR, 4-8
E_IC_SESSION_CLOSE, 4-9
E_IC_SESSION_EST, 4-10
E_IC_STATUS, 4-11
E_IC_STATUS_RESULT, 4-12
E_IC_XMT_DONE, 4-13
E_IC_XMT_ERROR, 4-14

XVT/Win API
accessory API, 2-3
general utilities, 2-9
memory management API, 2-6