System i

# Programming
# User Interface Manager APIs

*Version 6 Release 1*

System i

# Programming
# User Interface Manager APIs

*Version 6 Release 1*

**Note**

Before using this information and the product it supports, read the information in “Notices,” on page 105.

This edition applies to version 6, release 1, modification 0 of IBM i5/OS (product number 5761-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

# Contents

# User Interface Manager APIs

The user interface manager (UIM) APIs handle various aspects of the user interface, allowing your applications to display help, display a command line window, convert date and time formats, control keyboard buffering, display screens and pop-up windows, and to build screens.

These APIs allow an application developer to manipulate the user interface. These APIs are used in combination with variables, lists, and panel definitions in a panel group object. For more information about creating panel group objects, see the Application Display Programming  manual.

The UIM APIs perform a wide variety of tasks in the following categories:

- "UIM Application APIs" on page 2
- "UIM Display APIs" on page 10
- "UIM List APIs" on page 34
- "UIM Print APIs" on page 59
- "UIM Variable Pool APIs" on page 66

The UIM exit programs are:

- "Exit Program for an Action List Option or Pull-Down Field Choice" on page 69 adds, updates, or removes a list entry for the application when the action list option or pull-down choice action is a command string.
- "Exit Program for Application Formatted Data" on page 72 updates the data formatted by the application every time a panel is displayed, and returns control to the UIM through a normal return.
- "CALL Program for a Function Key" on page 75 is assigned to a function key to process requests that are specific to the application.
- "CALL Program for a Menu Item" on page 77 is assigned to a menu item to process requests that are specific to the application.
- "CALL Program for an Action List Option or Pull-Down Field Choice" on page 79 can specify the CALL dialog command on the ENTER, EXTENTER, PROMPT, and EXTPROMPT attributes of the list action (LISTACT) tag.
- "Exit Program for Conditioning Panel Items" on page 82 may be called during condition evaluation for conditions specified on the EXPR attribute of the COND tag and the LINKWHEN and UNLESSn attributes of the LINK tag.
- "Exit Program for a Cursor-Sensitive Prompt" on page 85 can be specified on the PROMPT attribute of the data item (DATAI), data item extender (DATAIX), and list column (LISTCOL) tags. This attribute specifies the name of a dialog variable identifying the program to call.
- "Exit Program for General Panel Checking" on page 88 may be specified on the USREXIT attribute of the panel definition (PANEL) tag. This attribute specifies the name of a dialog variable identifying the program to call.
- "Exit Program for an Incomplete List" on page 93 allows an application to display part of a list without having to build the entire list.
- "Exit Program for Text Area Data" on page 95 can update the data in a text area every time a panel is displayed and returns control to the UIM through a normal return.

For additional information, see:

- "Using the User Interface Manager APIs" on page 99
- "Using the User Interface Manager Exit Programs" on page 100

## APIs

These are the APIs for this category.

## UIM Application APIs

The UIM application APIs are:

- "Close Application (QUICLOA) API" (QUICLOA) closes a UIM application that was opened using the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API.
- "Open Display Application (QUIOPNDA) API" on page 3 (QUIOPNDA) initiates a UIM display application by opening the panel group that the application program specifies.
- "Open Print Application (QUIOPNPA) API" on page 6 (QUIOPNPA) initiates a UIM print application by opening the panel group that the application program specifies.

"User Interface Manager APIs," on page 1 | APIs by category

## Close Application (QUICLOA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Close option | Input | Char(1) |
| 3 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Close Application (QUICLOA) API closes a UIM application that was opened using the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API. The open and close APIs must be used in pairs to open and close each UIM application.

The QUICLOA API releases all UIM resources associated with the open application, destroying the variable pool and deleting any associated active lists. The storage for internal control blocks supporting the application is freed, and all locks acquired for the open application are released.

### Authorities and Locks

None.

### Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The open application for which this API is called. The application handle is assigned by the UIM and returned to the application program when the application is opened.

**Close option**
INPUT; CHAR(1)

Indicates whether to perform a normal or abnormal close operation. One of the following values must be specified:

*A* An abnormal close operation is performed. This operation is used when the application prematurely ends processing in the most efficient manner possible. When an application with an open printer file is closed with the abnormal option, the printer trailer text specified on the print trailer message (PRTTRAIL) tag is not printed.

*M* A normal close operation is performed.

**Error code**

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A11 E | Value is not correct. Reason code is &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Open Display Application (QUIOPNDA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Output | Char(8) |
| 2 | Qualified panel group name | Input | Char(20) |
| 3 | Application scope | Input | Binary(4) |
| 4 | Exit parameter interface | Input | Binary(4) |
| 5 | Full-screen help | Input | Char(1) |
| 6 | Error code | I/O | Char(*) |

Optional Parameter Group:

| | | | |
|---|---|---|---|
| 7 | Open data receiver | Output | Char(*) |
| 8 | Length of open data receiver | Input | Binary(4) |
| 9 | Length of available open data | Output | Binary(4) |

Default Public Authority: *USE
Threadsafe: No

The Open Display Application (QUIOPNDA) API initiates a UIM display application by opening the panel group that the application program specifies. The QUIOPNDA API and the Close Application (QUICLOA) API must be used in pairs to open and close each UIM display application.

Multiple applications can be opened at the same time. Each open application contains a complete set of dialog variables and active lists, and is independent of other open applications. A panel group can be opened more than once per job, but each call of the QUIOPNDA API initiates a new UIM display application and returns a unique application handle.

## Authorities and Locks

*Library Authority*
*READ

*Panel Group Authority*
*USE

*Panel Group Lock*
*SHRRD

## Required Parameter Group

**Application handle**
OUTPUT; CHAR(8)

The application handle for the opened application. The QUIOPNDA API returns a unique handle for each application opened. This handle must be provided as a parameter to every other UIM API that operates on the application, including the QUICLOA API, which must be used to close the application.

**Qualified panel group name**
INPUT; CHAR(20)

The name of the *PNLGRP object opened for this UIM application. The first 10 characters contain the name of the *PNLGRP object, and the second 10 characters contain the name of the library in which the panel group resides. These special values can be used for the library name:

| | |
|---|---|
| **CURLIB* | The job's current library |
| **LIBL* | The library list |

**Application scope**
INPUT; BINARY(4)

The scope of the resources for the application. The UIM uses the scope to determine whether or not to automatically close the application when reclaim resource processing is performed through the Reclaim Resource command (RCLRSC), the Reclaim Activation Group command (RCLACTGRP), or the End Request command (ENDRQS). During reclaim resource processing, the UIM closes all applications whose scope is no longer active.

One of the following values must be used:

| | |
|---|---|
| *-1* | The calling program is the scope for the application. If the calling program is an original program model (OPM) program and the application program is no longer active, the UIM will automatically close the application during reclaim resource processing. If the calling program is an Integrated Language Environment® (ILE) program and the activation group mark is no longer active, the UIM will automatically close the application during reclaim resource processing. |
| *0* | The job is the scope for the application. The application is not automatically closed by the UIM until the job is ended. |

**Exit parameter interface**
INPUT; BINARY(4)

The type of parameter interface used with exit programs and programs called as a result of the CALL dialog command for the UIM application being opened.

All exit and CALL programs receive a single parameter or multiple parameters, which are space pointers to information describing the state of the UIM application.

One of the following values must be used:

*0* Used by programs written in languages that can efficiently process structures.

*1* Used by programs written in languages that cannot efficiently process structures. This value indicates that all application programs called as exits or as a result of the CALL dialog command accept the parameter lists described for interface level 1.

*2* Used by programs written in languages that cannot efficiently process structures. This value indicates that all application programs called as exits or as a result of the CALL dialog command accept the parameter lists described for interface level 2.

For a detailed description of the structure passed for each type of exit and CALL program and for a description of the exit parameter lists, see "User Interface Manager APIs," on page 1.

**Full-screen help**
INPUT; CHAR(1)

Determines whether or not the UIM help for the application is displayed in pop-up windows or with a full screen. One of the following values must be used:

*Y* The online help information is displayed with a full screen.

*N* The online help information is displayed using pop-up windows, unless the user profile indicates that help is displayed with a full screen.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Optional Parameter Group

**Open data receiver**
OUTPUT; CHAR(*)

The receiver variable that is to receive the open data information requested. For the format of the open data receiver variable, see "Format of Data Returned."

**Length of open data receiver**
INPUT; BINARY(4)

The amount of data the application program is prepared to receive. If the length specified is larger than the amount of data available, the receiver is not changed beyond the amount of data available. If the length specified is greater than the actual length of the open data receiver parameter, unpredictable results may occur.

**Length of available open data**
OUTPUT; BINARY(4)

The length of all open data available. All available open data is returned if enough space is provided.

## Format of Data Returned

The format of the data available, returned in the open data receiver parameter, is as follows:

*CHAR(1)* Whether or not an error occurred while attempting to obtain the conversion tables needed to process the panel group. The conversion tables are needed when the CHRID attribute of the panel group is not equal to the CHRID attribute of the device, or when the CHRID attribute of the panel group is *JOBCCSID and the job CCSID is not equal to the device CHRID. A CPD6A2A diagnostic message will be logged in the job log for each conversion table that is not found.

One of the following values is returned:

*N* No error occurred while obtaining the conversion tables or the conversion tables were not necessary.

*Y* An error occurred while obtaining the conversion tables.

*CHAR(1)* Whether or not the conversion of data from the job to the device and from the device to the job will result in loss of fidelity of the data. Conversion will be done when the CHRID attribute of the panel group is not equal to the CHRID attribute of the device, or when the CHRID attribute of the panel group is *JOBCCSID and the job CCSID is not equal to the device CHRID.

One of the following values is returned:

*N* No loss of fidelity will occur.

*Y* Loss of fidelity may occur on the conversions.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A1E E | Object cannot be used with this device or print file. |
| CPF6A12 E | Unable to open panel group. |
| CPF6A17 E | Panel group &1 in library &2 is not at the current release level. |
| CPF6A2A E | Value for Application Scope parameter not valid. |
| CPF6A2E E | Value for Exit Parameter Interface parameter not valid. |
| CPF6A2F E | Value for Full-screen Help parameter not valid. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A26 E | Resources not available to open application. |
| CPF6A3A E | Value for Open Data Receiver is not valid. Reason code &1. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

# Open Print Application (QUIOPNPA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Output | Char(8) |
| 2 | Qualified panel group name | Input | Char(20) |
| 3 | Application scope | Input | Binary(4) |
| 4 | Exit parameter interface | Input | Binary(4) |
| 5 | Qualified name of printer device file | Input | Char(20) |
| 6 | Alternative file name | Input | Char(10) |
| 7 | Share open data path | Input | Char(1) |
| 8 | User data | Input | Char(10) |
| 9 | Error code | I/O | Char(*) |

Optional Parameter Group:

| | | | |
|---|---|---|---|
| 10 | Open data receiver | Output | Char(*) |
| 11 | Length of open data receiver | Input | Binary(4) |
| 12 | Length of available open data | Output | Binary(4) |

Default Public Authority: *USE
Threadsafe: No

The Open Print Application (QUIOPNPA) API initiates a UIM print application by opening the panel group that the application program specifies. The QUIOPNPA API and the Close Application (QUICLOA) API must be used in pairs to open and close each UIM print application.

Multiple applications can be opened at the same time. Each open application contains a complete set of dialog variables and active lists, and is independent of other open applications. The same panel group can be opened more than once per job, but each call of the QUIOPNPA API initiates a new UIM print application and returns a unique application handle.

## Authorities and Locks

*Library Authority*
*READ

*Panel Group Authority*
*USE

*Panel Group Lock*
*SHRRD

*Printer Device File Authority*
*USE

*Printer Device File Lock*
*SHRNUP

## Required Parameter Group

**Application handle**
OUTPUT; CHAR(8)

The application handle for the open application. The Open Display Application (QUIOPNDA) API returns a unique handle for each application opened. This handle must be provided as a parameter to every other UIM API that operates on the application, including the QUICLOA API, which must be used to close the application.

**Qualified panel group name**
INPUT; CHAR(20)

The name of the *PNLGRP object opened for this UIM application. The first 10 characters contain the name of the *PNLGRP object, and the second 10 characters contain the name of the library in which the panel group resides. These special values can be used for the library name:

| | |
|---|---|
| *CURLIB | The job's current library |
| *LIBL | The library list |

**Application scope**
INPUT; BINARY(4)

The scope of the resources for the application. The UIM uses the scope to determine whether or not to automatically close the application when reclaim resource processing is performed through the Reclaim Resource (RCLRSC) command, the Reclaim Activation Group (RCLACTGRP)

command, or the End Request (ENDRQS) command. During reclaim resource processing, the UIM closes all applications whose scope is no longer active.

One of the following values must be used:

*-1* The calling program is the scope for the application. If the calling program is an original program model (OPM) program and the application program is no longer active, the UIM will automatically close the application during reclaim resource processing. If the calling program is an Integrated Language Environment® (ILE) program and the activation group mark is no longer active, the UIM will automatically close the application during reclaim resource processing.

*0* The job is the scope for the application. The application is not automatically closed by the UIM until the job is ended.

**Exit parameter interface**

INPUT; BINARY(4)

The type of parameter interface used with exit programs and programs called as a result of the CALL dialog command for the UIM application being opened.

All exit and CALL programs receive a single parameter or multiple parameters, which are space pointers to information describing the state of the UIM application.

One of the following values must be used:

*0* Used by programs written in languages that can efficiently process structures.

*1* Used by programs written in languages that cannot efficiently process structures. This value indicates that all application programs called as exits or as a result of the CALL dialog command accept the parameter lists described for interface level 1.

*2* Used by programs written in languages that cannot efficiently process structures. This value indicates that all application programs called as exits or as a result of the CALL dialog command accept the parameter lists described for interface level 2.

For a detailed description of the exit parameter lists, and for a description of the type of structure passed for each type of exit and CALL program, see "User Interface Manager APIs," on page 1.

**Qualified name of printer device file**

INPUT; CHAR(20)

The name of the printer device file used in print operations. The first 10 characters contain the *FILE object name, and the second 10 characters contain the name of the library in which the printer device file resides. These special values can be used for the library name:

**CURLIB* The job's current library

**LIBL* The library list

**Alternative file name**

INPUT; CHAR(10)

An alternative name for the spooled file. The following special value can be used:

**NONE* There is no alternative name for the spooled file. The name of the spooled file is the name of the printer device file.

**Share open data path**

INPUT; CHAR(1)

Determines whether or not the printer device file's open data path (ODP) is shared. Sharing the ODP allows multiple UIM applications to print to the same spooled file. One of the following values must be used:

*Y* The ODP is shared.
*N* The ODP is not shared.
*F* Use the share value of the print file.

**User data**
INPUT; CHAR(10)

User data associated with the spooled file. This data becomes an attribute of the spooled file. The following special values can be used:

**FILE* The user data of the spooled file will be set to the user data value of the printer file being opened.
**NONE* There is no user data associated with the spooled file. The user data value of the printer file being opened will be set to blanks.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Optional Parameter Group

**Open data receiver**
OUTPUT; CHAR(*)

The receiver variable that is to receive the open data information requested. For the format of the open data receiver variable, see "Format of Data Returned."

**Length of open data receiver**
INPUT; BINARY(4)

The amount of data the application program is prepared to receive. If the length specified is larger than the amount of data available, the receiver is not changed beyond the amount of data available. If the length specified is greater than the actual length of the open data receiver parameter, unpredictable results may occur.

**Length of available open data**
OUTPUT; BINARY(4)

The length of all open data available. All available open data is returned if enough space is provided.

## Format of Data Returned

The format of the data available, returned in the open data receiver parameter, is as follows:

One of the following values is returned:

*CHAR(1)* Whether or not an error occurred while attempting to obtain the conversion tables needed to process the panel group. The conversion tables are needed when the CHRID attribute of the panel group is not equal to the CHRID attribute of the device, or when the CHRID attribute of the panel group is *JOBCCSID and the job CCSID is not equal to the device CHRID. A CPD6A2A diagnostic message will be logged in the job log for each conversion table that is not found.

*N* No error occurred while obtaining the conversion tables or the conversion tables were not necessary.

*Y* An error occurred while obtaining the conversion tables.

*CHAR(1)* Whether or not the conversion of data from the job to the device and from the device to the job will result in loss of fidelity of the data. Conversion will be done when the CHRID attribute of the panel group is not equal to the CHRID attribute of the device, or when the CHRID attribute of the panel group is *JOBCCSID and the job CCSID is not equal to the device CHRID.

One of the following values is returned:

*N* No loss of fidelity will occur.

*Y* Loss of fidelity may occur on the conversions.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A1C E | Unable to add print function. |
| CPF6A1E E | Object cannot be used with this device or print file. |
| CPF6A11 E | Value is not correct. Reason code is &3. |
| CPF6A12 E | Unable to open panel group. |
| CPF6A17 E | Panel group &1 in library &2 is not at the current release level. |
| CPF6A2A E | Value for Application Scope parameter not valid. |
| CPF6A2E E | Value for Exit Parameter Interface parameter not valid. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A26 E | Resources not available to open application. |
| CPF6A3A E | Value for Open Data Receiver is not valid. Reason code &1. |
| CPF9850 E | Override of printer file &1 not allowed. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## UIM Display APIs

The UIM display APIs are:

- "Add Pop-Up Window (QUIADDPW) API" on page 11 (QUIADDPW) begins the ability to display panels within a pop-up window.
- "Display Command Line Window (QUSCMDLN) API" on page 13 (QUSCMDLN) displays a window containing a command line.
- "Display Help (QUHDSPH) API" on page 14 (QUHDSPH) displays help information.
- "Display Long Text (QUILNGTX) API" on page 20 (QUILNGTX) displays a pop-up window containing the string of text that is passed to it.
- "Display Panel (QUIDSPP) API" on page 21 (QUIDSPP) displays a panel and waits for the user to press either a function key or the Enter key.
- "Remove Pop-Up Window (QUIRMVPW) API" on page 27 (QUIRMVPW) removes the pop-up window created by the Add Pop-Up Window (QUIADDPW) API.
- "Retrieve Help Text (QUHRHLPT) API" on page 28 (QUHRHLPT) lets you generate an Extended Markup Language (XML) source listing of the help identifiers in a panel group or menu object.
- "Set Screen Image (QUISETSC) API" on page 33 (QUISETSC) establishes the screen image for a UIM application.

"User Interface Manager APIs," on page 1 | APIs by category

# Add Pop-Up Window (QUIADDPW) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Pop-up window location | Input | Char(10) |
| 3 | Row | Input | Binary(4) |
| 4 | Column | Input | Binary(4) |
| 5 | Error code | I/O | Char(*) |

Optional Parameter Group:

| | | | |
|---|---|---|---|
| 6 | Depth | Input | Binary(4) |

Default Public Authority: *USE
Threadsafe: No

The Add Pop-Up Window (QUIADDPW) API begins the ability to display panels within a pop-up window. All subsequent panel displays for the open application remain in the pop-up window until the Remove Pop-up Window (QUIRMVPW) API is called, or until the QUIADDPW API is called again to add another pop-up window. The maximum number of pop-up windows that can be added to an open application is 20.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API when the application is opened.

**Pop-up window location**
INPUT; CHAR(10)

The name of a field used for field-adjacent positioning. Field-adjacent positioning places a pop-up window near a field on the underlying panel. The field name must correspond to a name specified on the NAME attribute on one of the following tags of the immediately underlying panel or pop-up window:

- Command Line (CMDLINE)
- Data Item Group (DATAGRP)
- Data Item (DATAI)
- List Column (LISTCOL)
- List Column Group (LISTGRP)
- Menu Item (MENUI)
- Option Line (OPTLINE)

One of these special values may be used to position the window:

| | |
|---|---|
| **OFFSET* | The pop-up window is offset from the top left corner of the most recent underlying panel or pop-up window. |
| **PULLDOWN* | The pop-up window is positioned adjacent to the previous position of the pull-down choice. If the most recent action for the most recently displayed panel was not the selection of a pull-down choice, offset positioning is used. |

*ROWCOL The row and column parameters indicate where the upper left corner of the pop-up window appears. Row and column positioning should be used only when displaying a pop-up window over a non-UIM panel, in conjunction with the Set Screen Image (QUISETSC) API.

**Row** INPUT; BINARY(4)

The absolute row used when row and column positioning is requested. The value is either a positive or negative integer.

If a negative integer is provided, the last row for the pop-up window is calculated using the absolute value of the integer, relative to the bottom of the display. A pop-up window cannot begin in row one, and a minimum-depth pop-up window must fit at the specified first row.

When a negative integer is provided, the absolute location of the window is determined by the size of the first panel displayed in the pop-up window. If another panel is displayed in the same window with a smaller width or depth, the location is not recalculated based on the negative integer. If another panel is displayed in the same window with a larger width or depth, the window is changed to use offset positioning.

**Column**
INPUT; BINARY(4)

The absolute column used when row and column positioning is requested. The value is either a positive or a negative integer.

If a negative integer is provided, the last column for the pop-up window is calculated using the absolute value of the integer, relative to the right side of the display. A pop-up window cannot begin in column one, and a minimum-width pop-up window must fit at the specified first column.

When a negative integer is provided, the absolute location of the window is determined by the size of the first panel displayed in the pop-up window. If another panel is displayed in the same window with a smaller width or depth, the location is not recalculated based on the negative integer. If another panel is displayed in the same window with a larger width or depth, the window is changed to use offset positioning.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Optional Parameter Group

**Depth** INPUT; BINARY(4)

The depth of the pop-up window to be displayed.

If the depth is specified on a panel without a TEXT area defined, it will be ignored and the depth specified on the PANEL tag will be used. If the depth specified on the API is larger than the depth specified on the PANEL tag, the depth specified on the API will be used.

The maximum supported depth for a pop-up panel depends on what the display device can support. On a device that can only support 24 rows by 80 bytes, the maximum depth of a pop-up window is 21 lines. On a device that can support 27 rows by 132 bytes, the maximum depth of a pop-up window is 24 lines.

The minimum allowed depth is 5 lines.

The special value of 0 may be specified to indicate that the depth on the PANEL tag should be used. The default value is 0.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0A E | Pop-up window depth too small. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A3E E | Application not open for display. |
| CPF6A81 E | Pop-up window cannot be added at this time. |
| CPF6A82 E | &4 is not a valid window location. |
| CPF6A83 E | Row or column given for positioning is not valid. |
| CPF6A85 E | Attempted to display more than &4 windows at a time. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Display Command Line Window (QUSCMDLN) API

Default Public Authority: *USE
Threadsafe: No

The Display Command Line Window (QUSCMDLN) API displays a window containing a command line. A **window** is an area on the display that is treated as a separate display. Windows have visible boundaries and appear to overlay the display from which they are requested.

From the command line displayed, the user can:

- Enter commands.
- Retrieve commands and run programs.
- Prompt for commands.
- Use help for CL commands.
- Request override commands. When the command line window is removed, all overrides are deleted.

## Authorities and Locks

None.

## Parameters

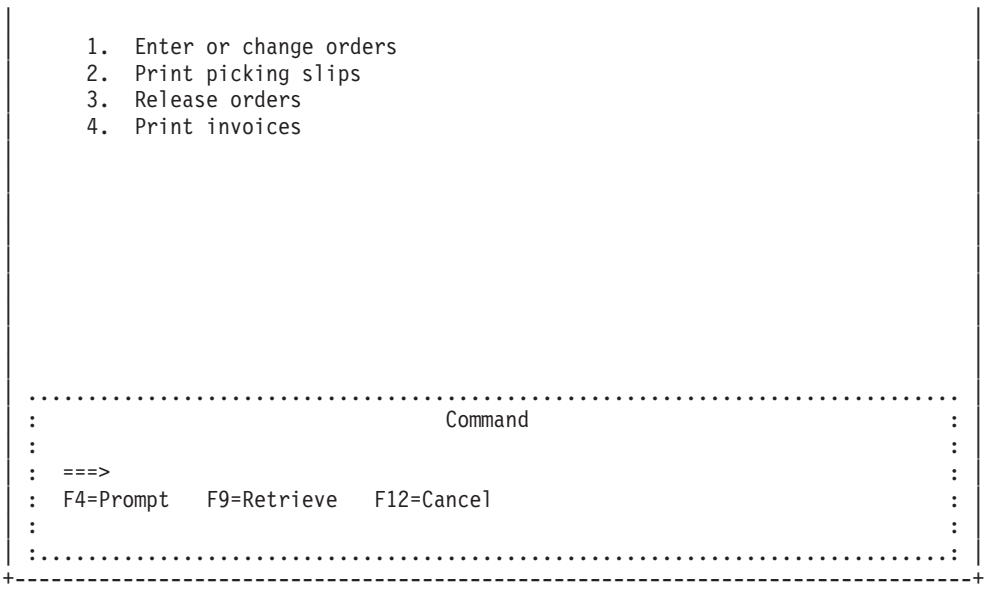
None.

## Usage Notes

For consistency with other i5/OS® displays, you should use the F9 key to request a command line window.

The command line window produced by the QUSCMDLN API is shown at the bottom of the following display:

```
+------------------------------------------------------------------------------+
| CSTINV                    Customer Invoice Menu                              |
|                                                                              |
| Select one of the following:                                                 |
```




## Display Appearance Problems

There are two rare situations where portions of the underlying display might not be shown correctly while the command line window is shown. The underlying display is not actually changed, and it is shown correctly once the command line window is removed.

The situations are:

1. The underlying display is a text editor or other display that uses special work station printer mode functions. While the command line window is displayed, the special work station printer mode characters are not shown.
2. The underlying display is a bidirectional right-to-left display. While the command line window is displayed, the underlying display is flipped and shown from left-to-right.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF9871 E | Error occurred while processing. |

API Introduced: V1R3

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Display Help (QUHDSPH) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Help identifier array | Input | Char(*) |
| 2 | Number of help identifiers | Input | Binary(4) |
| 3 | Help type | Input | Array(2) of Binary(4) |
| 4 | Full display title | Input | Char(55) |
| 5 | Qualified search index object name | Input | Char(20) |
| 6 | Display type | Input | Char(1) |
| 7 | Upper-left corner | Input | Array(2) of Binary(4) |
| 8 | Lower-right corner | Input | Array(2) of Binary(4) |
| 9 | Cursor location | Input | Array(2) of Binary(4) |

| | | | |
|---|---|---|---|
| 10 | Error code | I/O | Char(*) |

Optional Parameter Group:

| | | | |
|---|---|---|---|
| 11 | Help bookshelf name | Input | Char(8) |
| 12 | Border attribute | Input | Array(2) of Char(1) |
| 13 | Border characters | Input | Array(8) of Char(1) |

Default Public Authority: *USE
Threadsafe: No

The Display Help (QUHDSPH) API displays help information. The help can be either contextual or extended. It can be formatted as a full display or in a window.

**Contextual help** provides information about a single item, such as the field on which the user's cursor is positioned when help is requested. **Extended help** provides help for all the items on the display; it contains all contextual help items and can contain additional information as well. A **window** is an area on the display that is treated as a separate display. Windows have visible boundaries and appear to overlay the display from which they are requested.

You can use the QUHDSPH API to handle Help key processing in applications written in data description specifications (DDS) or user-defined data streams (UDDS). You do not need it to handle help requests in applications that use record-level DDS display files with DDS help keywords to present panels because the DDS help keywords handle all help requests.

You do not need to use this API to display help for applications written using panel groups to present panels, because the UIM handles all help requests.

## Authorities and Locks

*Library Authority*
*READ

*Search Index Object Authority*
*USE

*Search Index Object Library Authority*
*READ

*Search Index Object Lock*
*SHRRD

*Panel Group Authority*
*USE

*Panel Group Lock*
*SHRRD

## Required Parameter Group

**Help identifier array**
INPUT; CHAR(*)

An array of the help identifiers to display. The list can contain up to 2000 items. Each item has two parts:

**Qualified help panel group name**
CHAR(20)

The panel group (*PNLGRP) object that contains the help module to display, and the library in which it is located. (A **panel group** is an object with an object type of *PNLGRP. It contains display panels, print panels, or help modules.)

The first 10 characters contain the panel group object name, and the second 10 characters contain the library name. You can use these special values for the library name:

*CURLIB The job's current library
*LIBL The library list

**Help module name**
CHAR(32)

The name specified on the NAME attribute of a :HELP. tag in the panel group object. The name must be specified using uppercase, alphabetic characters.

**Number of help identifiers**
INPUT; BINARY(4)

The number of help identifiers in the help identifier array parameter. The number must be between 1 and 2000.

**Help type**
INPUT; ARRAY(2) of BINARY(4)

Whether this is a request to display extended or contextual help, and which help identifiers to display for the latter. For contextual help, only a subset of the help identifiers in the help identifier array parameter is initially displayed, and a function key is enabled to display extended help. Extended help includes all help identifiers in the help identifier array parameter, including those used in contextual help.

This parameter is a 2-element array of BINARY(4) values. Both elements are used as indexes into the help identifier array. The array elements are:

1. The first help identifier to display for contextual help. The value must be greater than or equal to 1, and less than or equal to the number of help identifiers in the help identifier array.
2. The last help identifier to display for contextual help. The value must be greater than or equal to the value specified in the first element of this parameter, and less than or equal to the number of help identifiers in the help identifier array.

To display extended help, set the value of the first array element to 1 and the value of the second array element to the value specified in the number of help identifiers parameter.

**Full display title**
INPUT; CHAR(55)

The default title to use if the help is shown in a full display and if no title is found in the help panel group object.

**Qualified search index object name**
INPUT; CHAR(20)

The search index (*SCHIDX) object that can be accessed from the help display, and the library in which it is located. The first 10 characters contain the search index object name, and the second 10 characters contain the library name.

You can use the following special value for this parameter:

*NONE The search index function is not made available for this help request. The library name must be blank.

You can use the following special values for the library name:

**CURLIB* The job's current library
**LIBL* The library list

**Note:** The qualified search index object name must be *NONE if the help bookshelf name is something other than *NONE.

**Display type**
INPUT; CHAR(1)

Whether the help information is displayed in a full screen or a window. You must use one of the following values:

*Y* The help is displayed in a full screen.
*N* The help can be displayed in a window, depending on the user option (USROPT) value of the user profile.

**Upper-left corner**
INPUT; ARRAY(2) of BINARY(4)

The upper-left corner of the area on the display for which help is requested. If the help is displayed in a window, the window is adjacent to the area identified, if possible. The array elements are:

- The row number of the upper-left corner
- The column number of the upper-left corner

**Lower-right corner**
INPUT; ARRAY(2) of BINARY(4)

The lower-right corner of the area on the display for which help is requested. If the help is displayed in a window, the window is adjacent to the area identified, if possible. The array elements are:

- The row number of the lower-right corner
- The column number of the lower-right corner

**Cursor location**
INPUT; ARRAY(2) of BINARY(4)

The position of the cursor when help is requested. If the help is displayed in a window, this cursor position is used by the UIM to decide the position and size of the window. The array elements are the row number of the cursor position and the column number of the cursor position.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Optional Parameter Group

**Help bookshelf name**
INPUT; CHAR(8)

This parameter is no longer supported and any value entered will be ignored.

**Border attribute**
INPUT; ARRAY(2) of CHAR(1)

The border attributes of the pop-up window in which the help may be displayed. The two array elements are:

- The border attribute for a monochrome display. The default for a monochrome display is normal.
- The border attribute for a color display. The default for a color display is blue.

**Border attributes**

| Hex Value | Color Display | Monochrome Display |
|---|---|---|
| X'01' | Green | Normal |
| X'02' | Green<br>Reverse | Reverse Image |
| X'03' | White | High Intensity |
| X'04' | Green<br>Reverse | High Intensity Reverse |
| X'05' | Green<br>Underscore | Underscore |
| X'06' | Green Underscore<br>Reverse | Underscore<br>Reverse |
| X'07' | White<br>Underscore | High Intensity<br>Underscore |
| X'08' | Nondisplay | Nondisplay |
| X'09' | Red | Blink |
| X'0A' | Red<br>Reverse | Blink<br>Reverse |
| X'0B' | Red<br>Blink | Blink<br>High Intensity |
| X'0C' | Red<br>Blink<br>Reverse | Blink<br>High Intensity<br>Reverse |
| X'0D' | Red<br>Underscore | Blink<br>Underscore |
| X'0E' | Red<br>Underscore<br>Reverse | Blink<br>Underscore<br>Reverse |
| X'0F' | Red<br>Blink<br>Underscore | Blink<br>Underscore<br>High Intensity |
| X'10' | Turquoise<br>Column Separator | Column Separator |
| X'11' | Turquoise<br>Column Separator<br>Reverse Image | Column Separator<br>Reverse Image |
| X'12' | Yellow<br>Column Separator | Column Separator<br>High Intensity |
| X'13' | Yellow<br>Column Separator<br>Reverse Image | Column Separator<br>High Intensity<br>Reverse Image |
| X'14' | Turquoise<br>Column Separator<br>Underscore | Column Separator<br>Underscore |

| Hex Value | Color Display | Monochrome Display |
|---|---|---|
| X'15' | Turquoise<br>Column Separator<br>Reverse Image<br>Underscore | Column Separator<br>Reverse Image<br>Underscore |
| X'16' | Yellow<br>Column Separator<br>Underscore | Column Separator<br>Underscore |
| X'17' | Pink | Blink<br>Column Separator |
| X'18' | Pink<br>Reverse Image | Blink<br>Column Separator<br>Reverse Image |
| X'19' | Blue | Blink<br>Column Separator<br>High Intensity |
| X'1A' | Blue<br>Reverse Image | Blink<br>Column Separator<br>High Intensity<br>Reverse Image |
| X'1B' | Pink<br>Underscore | Blink<br>Column Separator<br>Underscore |
| X'1C' | Pink<br>Reverse Image<br>Underscore | Blink<br>Column Separator<br>Reverse Image<br>Underscore |
| X'1D' | Blue<br>Underscore | Blink<br>Column Separator<br>High Intensity<br>Underscore |

**Border characters**

INPUT; ARRAY(8) of CHAR(1)

The border characters that define the pop-up window in which the help may be displayed. The eight array characters are:

1. The character to define the top left corner.
2. The character to define the top margin.
3. The character to define the top right corner.
4. The character to define the left margin.
5. The character to define the right margin.
6. The character to define the bottom left corner.
7. The character to define the bottom margin.
8. The character to define the bottom right corner.

The help window would look like this:

```
12222222222223
4             5
4             5
4             5
67777777777778
```

When using the default values, the border would look like this:

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6Exx E | All CPF6Exx messages could be signalled. xx is from 01 to FF. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R1

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Display Long Text (QUILNGTX) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Text string | Input | Char(*) |
| 2 | Length of text string | Input | Binary(4) |
| 3 | Message ID | Input | Char(7) |
| 4 | Qualified message file name | Input | Char(20) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Display Long Text (QUILNGTX) API displays a pop-up window containing the string of text that is passed to it.

This API may not be used to display text that is bidirectional right to left.

## Authorities and Locks

None.

## Required Parameter Group

**Text string**
INPUT; CHAR(*)

The text string that is to be displayed in a pop-up window.

**Length of text string**
INPUT; BINARY(4)

The length of the text string, in bytes. The value must be greater than zero and less than or equal to 15 728 640.

**Message ID**
INPUT; CHAR(7)

The specified message ID of the panel title text that will be retrieved. If the message ID is not found in the specified message file, the message ID will be displayed at the top of the panel as the title. If the message ID is blank, the title for the panel is blank.

**Qualified message file name**
INPUT; CHAR(20)

The name of the message file containing the message ID. The first 10 characters contain the name of the *MSGF object, and the second 10 characters contain the name of the library in which the message file resides. If the message file and library name are left blank, the program uses the default QCPFMSG message file in *LIBL. You can use the special value *LIBL for the library name.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed |
| CPF6A4C E | At least one parameter value is not correct. Reason code is &1 |
| CPF9871 E | Error occurred while processing |

API introduced: V3R6

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

## Display Panel (QUIDSPP) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Function requested | Output | Binary(4) |
| 3 | Panel name | Input | Char(10) |
| 4 | Redisplay option | Input | Char(1) |
| 5 | Error code | I/O | Char(*) |

Optional Parameter Group 1:

| | | | |
|---|---|---|---|
| 6 | User task | Input | Char(1) |
| 7 | Call stack counter | Input | Binary(4) |
| 8 | Call message queue | Input | Char(*) |
| 9 | Message reference key | Input | Char(4) |
| 10 | Cursor position option | Input | Char(1) |
| 11 | Last list entry | Input | Char(4) |
| 12 | Error list entry | Input | Char(4) |
| 13 | Wait time | Input | Binary(4) |

Optional Parameter Group 2:

| | | | |
|---|---|---|---|
| 14 | Length of call message queue name | Input | Binary(4) |
| 15 | Call qualification | Input | Char(20) |

Default Public Authority: *USE
Threadsafe: No

The Display Panel (QUIDSPP) API displays a panel and waits for the user to press either a function key or the Enter key.

The values for all I/O fields in the panel definition are taken from dialog variables in the variable pool. If the panel contains list areas, these values are also taken from list entries in the lists associated with the open application.

The application program can also control which list entry is initially displayed at the top of a list area by using the Set List Attributes (QUISETLA) API. If the panel contains a list area displaying an incomplete list and if the list does not have enough entries to fill the list area, the UIM calls the program identified by the program dialog variable parameter of the QUISETLA API to add more list entries. The program is called repeatedly until the requested number of entries is added or until the list is marked as complete.

If the panel contains an extended action list area and if the list is not currently active in the open application, the list is activated by the QUIDSPP API.

Any information the user enters into input fields on the panel is validated according to the specifications of the tag language, then saved in the corresponding dialog variables and list entries.

Control returns to the application program only after all necessary functions are completed as defined by UIM processing rules and tag language specifications. The application program can retrieve dialog variables and list entries after the QUIDSPP API returns to determine what values were specified by the user.

The QUIDSPP API can be called with a long or short argument list. For the short argument list, arguments must be passed to the QUIDSPP API with the required parameters described below. For the long argument list, arguments must be passed to the QUIDSPP API with all the required and optional parameters described below.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API when the application is opened.

**Function requested**
OUTPUT; BINARY(4)

The function requested by the user. Only special UIM-defined functions and functions using the RETURN dialog command are returned to the application program. Any other functions requested by the user are either handled by the UIM or rejected. If they are rejected, an error message is displayed on the panel.

The RETURN dialog command can be specified as follows:

- On the ACTION attribute of the key list item (KEYI), pull-down field choice (PDFLDC), or menu item (MENUI) UIM tags
- On the ENTER and SELECT attributes of the display panel (PANEL) UIM tag

The RETURN dialog command returns numbers from 1 through 32 767. The return values for UIM-defined functions are as follows:

*0* The enter function is requested. This value is returned in the following situations:

- When the user has entered options in the list for a panel with a list area of ACTOR=CALLER.
- When enter has been pressed in a text area and the text area exit program has sent message CPF6A07 to end the enter function and return to the application program.

*-4* The exit function is requested.
*-8* The cancel function is requested.
*-10* The prompt function is requested. This value is returned only for a panel with an ACTOR=CALLER list area, and only when the user has entered options in the list.
*-20* No function was requested before the time specified on the wait time parameter has ended. This value is returned only when the wait time parameter specifies a time-out value.

**Panel name**
INPUT; CHAR(10)

The name of the panel to display. The panel must be defined in the panel group for the open application.

**Redisplay option**
INPUT; CHAR(1)

Indicates whether or not the panel is formatted and displayed as a first-time display or as a redisplay.

One of the following values must be used:

*Y* The panel is formatted as a redisplay. If the panel has not been displayed in the application before, processing for the first-time display is done and this parameter is ignored.

Cursor positioning that is controlled by the program is ignored when the redisplay option is used. Cursor positioning controlled by action list processing is supported when the redisplay option is used.

Dialog variables are edited for a new displayable value only if the variable pool contains a value more current than the value from the last time the panel was displayed. When VARUPD=NO is processed, the panel is redisplayed with values entered in input fields but not stored in the variable pool.

*N* The panel is formatted as a first-time display. The panel is displayed with the first conditioned-on item at the top of each pageable data and menu area. Pageable information areas are positioned at the first line of text. All dialog variables displayed on the screen are edited to produce a displayable value.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Optional Parameter Group 1

**User task**
INPUT; CHAR(1)

Indicates whether or not this panel is the logical start of a new user task. The value used determines whether or not the UIM returns to the application program when the EXIT dialog command is requested by a lower level program, such as an action on the KEYI tag.

The user task has no effect on UIM operations when the EXIT dialog command is specified directly in the definition of this panel as the function key, pull-down choice, or menu item action.

One of the following values must be specified:

*N* This panel is the logical start of a new user task. If the user generates the EXIT dialog command at a lower call level (for example, by pressing the exit function key on a panel displayed by a list action from the original panel), this panel is redisplayed without returning control to the application program.

*O* This panel is not the start of a new user task. If the user generates the EXIT dialog command at a lower call level, control returns to the application program with an indication that the exit function was requested. This is the default value when a short argument list is passed to the QUIDSPP API.

**Call stack counter**
INPUT; BINARY(4)

A number identifying the location in the program stack of the call message queue for the UIM to use. This parameter is used with the call message queue parameter.

Any nonnegative number can be specified for the relative program call number.

*0* The message queue of the program specified in the call message queue parameter. This is the default value when a short argument list is passed to the QUIDSPP API.

*1* The message queue of the program that calls the program specified in the call message queue parameter is used for messages displayed to the user.

*n* The message queue of the *n*th program up the stack from the program specified in the call message queue parameter is used for messages displayed to the user.

**Call message queue**
INPUT; CHAR(*)

The name of the message queue for the UIM to use, or the name of the program to start counting from when using a value other than zero for the call stack counter parameter. The program you specify must be in the call stack. The UIM will:

- Receive a message from this queue to initially display on the message line of the panel. This is done using the value passed for the message reference key parameter.
- Move to this message queue all completion, information, diagnostic, and escape messages sent to the UIM by programs or commands called to process an action defined in the panel. Note that escape messages are changed to diagnostic messages when they are moved.

The following special value can be used:

**CALLER* The application program calling the QUIDSPP API is the starting point for identifying the call message queue. This is the default value when a short argument list is passed to the QUIDSPP API.

This parameter can be from 1 to 256 characters in length. The length of call message queue parameter specifies this parameter's length. If the length of call message queue parameter is not used, then the length of this parameter is assumed to be 10 characters. If further qualification is needed to identify the call, the call qualification parameter can be used.

**Message reference key**
INPUT; CHAR(4)

The messages in the call message queue that are displayed on the panel. The call message queue is identified by the call stack counter parameter and the call message queue parameter. This parameter must be set to the message reference key for the first (oldest) message on the call message queue that should be displayed on the panel. One of the following special values can be used:

*(blank)* No messages are shown to the user on the initial panel, even if there are messages in the call message queue. This is the default value when a short argument list is passed to the Display Panel (QUIDSPP) API.

*FRST* All new messages in the call message queue, starting with the first new message in the queue, are presented on the initial panel.

**Cursor position option**
INPUT; CHAR(1)

The rules that determine the initial position of the cursor when the panel is displayed. This parameter controls only the initial cursor placement when the panel is displayed by the QUIDSPP API. When a panel is redisplayed by the UIM or by the application program, the cursor remains where it was left by the user.

One of the following values must be used:

*A* Cursor positioning for action list processing is performed. This value is used when the panel is redisplayed after action list processing when ACTOR=CALLER is on the list area (LIST) tag. The cursor and list are positioned as defined by the last list entry and error list entry parameters.

*D* Default cursor positioning is performed. The specific cursor position depends on whether or not any variables or list entries associated with the panel are marked in error. This is the default value when a short argument list is passed to the display panel (QUIDSPP) API.

*P* Cursor positioning that is controlled by the program is performed. The cursor is positioned using the current value of the dialog variables associated with the CSRVAR, CSRPOS, CSRLST, and CSREID attributes of the panel definition (PANEL) tag. Cursor positioning controlled by the program is allowed only for a panel containing all four cursor positioning attributes.

With cursor positioning controlled by the program, each pageable, nonlist area is repositioned to make sure that the first input field in error is visible, but the cursor is not positioned at the first input field in error. If the value of any of the dialog variables for cursor position identified in the panel definition is unusable, the cursor defaults to the first input field on the panel.

**Last list entry**

INPUT; CHAR(4)

The list entry handle for the last action list entry that had an action processed.

This information is used to position the cursor. Cursor positioning for action list processing must be specified in the cursor position option parameter or this parameter is ignored. One of the following special values can be used:

*EXTE* The last list action processed is for the extended action entry.

*NONE* The application is providing no value for this parameter. This is the default value when a short argument list is passed to the Display Panel (QUIDSPP) API.

**Error list entry**

INPUT; CHAR(4)

The list entry handle for the first and only list entry that had an error while processing list actions. This parameter is also set when action list processing is stopped because a user pressed F3 (Exit) or F12 (Cancel). This information displays the correct page of list information.

Cursor positioning for action list processing must be specified in the cursor position option parameter, or this parameter is ignored.

One of the following special values can be used:

*EXTE* The extended action entry is in error; the UIM does not reposition the list.

*NONE* The application is providing no value for this parameter. This is the default value when a short argument list is passed to the Display Panel (QUIDSPP) API.

**Wait time**

INPUT; BINARY(4)

The number of seconds to wait for data to become available from the work station. After this amount of time, the keyboard locks, control returns to the application, and the function requested parameter returns an indication that the wait time ended. Because the wait time ended, input dialog variables associated with the panel are not updated. The number of seconds specified should be a positive integer.

One of the following special values can be used:

*-1* The UIM waits indefinitely until data becomes available from the work station. This is the default value when a short argument list is passed to the Display Panel (QUIDSPP) API.

*0* The UIM does not wait for data to become available from the work station. The panel is displayed with the keyboard locked, and control returns immediately to the application with an indication in the function requested parameter that the wait time ended.

## Optional Parameter Group 2

**Length of call message queue name**
INPUT; BINARY(4)

The length of the call message queue name. Valid values for this parameter range from 1 to 256. If the value is not valid, an error will occur.

The default value for this parameter is 10.

**Call qualification**
INPUT; CHAR(20)

The name of the module and bound program that contain the procedure. This value is used to further qualify the procedure identified in the call message queue parameter. The first 10 characters specify the module name, and the second 10 characters specify the bound program name. When the call qualification parameter is specified, the call stack will be searched only for a procedure within a bound program.

If this parameter is not used, only the call message queue parameter will be used to identify the call.

The special value of *NONE can be used for the module or bound program name, or for both. When *NONE is specified for one of the names, then that name will not be used when searching for the qualified procedure. If *NONE is specified for both names, only the call message queue parameter will be used to identify the call. Original program model (OPM) programs should specify *NONE for both names.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C36 E | Number of parameters, &1, entered for this API was not valid. |
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A11 E | Value is not correct. Reason code is &3. |
| CPF6A13 E | Application &3 closed prematurely. |
| CPF6A14 E | Program defined by variable &4 cannot be called. |
| CPF6A15 E | Errors occurred in list exit program. |
| CPF6A22 E | Panel cannot be displayed in a window. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A3E E | Application not open for display. |
| CPF6A3F E | Panel &4 was not found in panel group &1. |
| CPF6A4A E | Panel &4 is too large to display as a pop-up window. |
| CPF6A4B E | Value for Redisplay Option parameter not valid. |
| CPF6A40 E | Panel &4 is already in use. |

| Message ID | Error Message Text |
|---|---|
| CPF6A41 E | Default cursor positioning required for panel &4. |
| CPF6A42 E | Display of panel &4 not allowed. |
| CPF6A43 E | Cannot use program message queue &6. |
| CPF6A50 E | Error was found during display file or printer file operation. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Remove Pop-Up Window (QUIRMVPW) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Remove option | Input | Char(1) |
| 3 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Remove Pop-Up Window (QUIRMVPW) API removes the pop-up window created by the Add Pop-Up Window (QUIADDPW) API. After calling the QUIRMVPW API, subsequent calls to the Display Panel (QUIDSPP) API display either a full-screen panel or a panel in a pop-up window defined by a previous call of the QUIADDPW API. A pop-up window cannot be removed if an action or exit program is currently being processed for a panel displayed in that pop-up window.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API when the application is opened.

**Remove option**
INPUT; CHAR(1)

The pop-up windows that are removed from the open application. One of the following values must be used:

*I* All inactive pop-up windows. A pop-up window is inactive if there are no panels currently displayed in that window.

*L* The most recently added pop-up window.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A3E E | Application not open for display. |
| CPF6A84 E | Window cannot be removed at this time. |
| CPF6A90 E | Value not correct. Reason code &3. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Retrieve Help Text (QUHRHLPT) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Receiver variable | Output | Char(*) |
| 2 | Length of receiver variable | Input | Binary(4) |
| 3 | Format name | Input | Char(8) |
| 4 | Help identifiers to be retrieved | Input | Array(*) of (*) |
| 5 | Number of help identifiers | Input | Binary(4) |
| 6 | Help identifier text | Output | Char(*) |
| 7 | Length of help identifier text | Input | Binary(4) |
| 8 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Help Text (QUHRHLPT) API lets you generate an Extensible Markup Language (XML) source listing of the help identifiers in a panel group or menu object. The QUHRHLPT API returns the XML source in UTF-8 (CCSID 1208). You can use the QUHRHLPT API to generate XML source representing the help text as defined in help identifiers in UIM panel groups or menus.

The QUHRHLPT API returns information in one format.

**Special consideration:** The text for the UIM LINK tag will be displayed, but there will be no link generated in the XML source.

## Authorities and Locks

*Help Panel Group/Menu Authority*
*READ

*Help Panel Group/Menu Library Authority*
*EXECUTE

*Imported Help Panel Group Authority*
*READ

*Imported Help Panel Group Library Authority*
*EXECUTE

## Required Parameter Group

**Receiver variable**
INPUT; CHAR(*)

The variable that is to receive the information requested. For the format of the structure, see "Format of the Receiver Variable."

**Length of receiver variable**
INPUT; BINARY(4)

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

**Format name**
INPUT; CHAR(8)

The format of the information returned for the help identifiers that are requested. The following format name must be used:

*RHLP0100*

For details about the formats, see "Format of the Receiver Variable."

**Help identifiers to be retrieved**
INPUT; Array(*) of (*)

An array of help identifiers to be formatted. The list can contain up to 2000 items. For the format of the structure, see "Format of Help Identifiers to Be Retrieved" on page 31.

**Number of help identifiers**
INPUT; BINARY(4)

The number of help identifer names passed in on the help identifiers to be retrieved parameter.

**Help identifier text**
OUTPUT; Char(*)

The variable that is to receive the generated Xml output. The information returned potentially could be very large. In this case, a user space with the auto-extendable attribute set may act as a better choice for a receiver variable. For further information about using a user space, see User spaces. For the format of the structure, see "Format of the Help Identifier Text" on page 31. If the variable is not large enough, no data may be returned.

**Length of help identifier text**
INPUT; BINARY(4)

The length of the help identifier text parameter. If the length specified is larger than the size of the parameter, the results may be undesirable. The minimum length is 8 bytes.

**Error code**
I/O; CHAR(*)

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

## Format of the Receiver Variable

The receiver variable contains:

- A header section
- A list data section

For detailed descriptions of each field in the header section, see "Field Descriptions" on page 30.

### RHLP0100

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4 | Bytes returned |
| 4 | 4 | BINARY(4) | Bytes available |
| 8 | 8 | CHAR(8) | Format name |
| 16 | 10 | BINARY(4) | Offset to help identifiers |
| 20 | 14 | BINARY(4) | Number of entries |
| 24 | 18 | BINARY(4) | Size of each entry |
| * | * | CHAR(*) | Help identifiers |

### Help Identifier Entries

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | CHAR(32) | Help identifier name |
| 32 | 20 | CHAR(10) | Object name |
| 42 | 2A | CHAR(10) | Object library name |
| 52 | 34 | CHAR(10) | Object type |
| 62 | 3E | CHAR(1) | Help identifier found |
| 63 | 3F | CHAR(96) | Help identifier anchor name |
| 167 | A7 | CHAR(33) | Reserved |

## Field Descriptions

**Bytes available.** The number of bytes of information available to be returned.

**Bytes returned.** The number of bytes of information returned.

**Format name.** The format name that was passed to this API.

**Help identifier anchor name.** The name of the anchor within the help document.

**Help identifier found.** A one-character field indicating the status of the help ID when the caller selects the help identifiers to be retrieved. The possible values are:

*0* Not found due to incorrect name
*1* Help identifier found
*2* Object access failure; see joblog for details

If none of the requested help identifiers are found, the API will also signal a CPF6E3B escape message.

**Help identifier name.** The name specified on the NAME attribute of a :HELP. tag in the panel group or menu object.

**Number of entries.** The number of entries being returned in the list data section.

**Object library name.** The library containing the panel group or menu object in which the help identifier was found.

**Object name.** A 10-character name used to identify the panel group or menu object in which the help identifier was found.

**Object type.** The type of the object in which the help identifier was found.

**Offset to help identifiers.** The offset from the start of the receiver variable to the first help identifier.

**Reserved.** An ignored field.

**Size of each entry.** The size of each entry in bytes in the list data section.

## Format of Help Identifiers to Be Retrieved

The following information is accepted as input. For detailed descriptions of the fields in the table, see "Field Descriptions."

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | CHAR(32) | Help identifier name |
| 32 | 20 | CHAR(10 | Object name |
| 42 | 2A | CHAR(10) | Object library name |
| 52 | 34 | CHAR(10) | Object type |
| 62 | 3E | CHAR(18) | Reserved |

## Field Descriptions

**Help identifier name.** The name specified on the NAME attribute of a :HELP. tag in the panel group or menu object. The name must be specified using uppercase, alphabetic characters.

**Object library name.** The library containing the panel group or menu object in which the help identifier was found. As input, this value must be in uppercase letters. You can use these special values for the library name:

**CURLIB* The job's current library
**LIBL* The library list

**Object name.** A 10-character name used to identify the panel group or menu object in which the help identifier was found. As input, this value must be in uppercase letters.

**Object type.** The type of the object in which the help identifier was found. This value must be *PNLGRP or *MENU.

**Reserved.** This field should be set to blanks.

## Format of the Help Identifier Text

The following is a description of the data in the help identifier text parameter.

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Bytes returned |
| 4 | 4 | BINARY(4) | Bytes available |
| * | * | Char(*) | Data |

## Field Descriptions

**Bytes available.** The number of bytes of information available to be returned.

**Bytes returned.** The number of bytes of information returned.

**Data.** The UIM online help formatted in Xml. This information contains:

- A document header section
- Xml-formatted tags
- A document footer section

Each formatted help identifier contains a  tag representing the help identifier being formatted, such as:

-  <a name="FSDIR.CRTDIR"></a>
- (Xml-formatted help text is contained here)
- 

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF3CF2 E | Error(s) occurred during running of &1 API. |
| CPF3C20 E | Error found by program &1. |
| CPF3C21 E | Format name &1 is not valid. |
| CPF3C24 E | Length of receiver variable not valid. |
| CPF3C31 D | Object type &1 is not valid. |
| CPF3C36 E | Number of parameters, &1, entered for this API was not valid. |
| CPF9801 D | Object &2 in library &3 not found. |
| CPF9802 D | Not authorized to object &2 in &3. |
| CPF6E3B E | Help information is not available. |
| CPF9803 D | Cannot allocate object &2 in library &3. |
| CPF9804 D | Object &2 in library &3 damaged. |
| CPF9810 D | Library &1 not found. |
| CPF9820 D | Not authorized to use library &1. |
| CPF9838 E | User profile storage limit exceeded. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V5R1

Top | "User Interface Manager APIs," on page 1 | APIs by category

# Set Screen Image (QUISETSC) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Set Screen Image (QUISETSC) API establishes the screen image for a UIM application. This API is used when a pop-up window is displayed over a panel that was not displayed using the same UIM application.

For displaying pop-up windows over data description specifications (DDS) screens, it is the responsibility of the application program to save and restore the display before and after displaying the pop-up window. This can be done by specifying RSTDSP(*YES) for the DDS display file.

When the QUISETSC API is called, a read screen command is sent to the requesting program device. The screen image returned from the read screen operation is saved for the open application. Subsequent panels displayed in pop-up windows are displayed on top of this image. Using this API to set the screen image is functionally similar to using the Display Panel (QUIDSPP) API to establish the primary panel on which pop-up windows are overlaid.

The UIM does not preserve the extended character buffer (ECB) attributes on the underlying panel. This means that the UIM issues a read screen command instead of a read screen with ECB command. Any highlighting on the underlying panel that was specified using ECB attributes is lost. When the application redisplays the underlying panel after the pop-up window is removed, the correct highlighting is restored.

No half-index ECB attributes are preserved.

Any double-byte character string (DBCS) data in the underlying panel is supported, including DBCS data whose shift-in and shift-out characters are specified in ECB attributes.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API when the application is opened.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |

| Message ID | Error Message Text |
|---|---|
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A1F E | An active display already exists for this application. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A3E E | Application not open for display. |
| CPF6A50 E | Error was found during display file or printer file operation. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## UIM List APIs

The UIM list APIs are:

- "Add List Entry (QUIADDLE) API" (QUIADDLE) adds one new entry to a list.
- "Add List Multiple Entries (QUIADDLM) API" on page 36 (QUIADDLM) adds one or more new entries to a list.
- "Delete List (QUIDLTL) API" on page 40 (QUIDLTL) deletes an active list and provides a way for the application to start over with a new list.
- "Get List Entry (QUIGETLE) API" on page 41 (QUIGETLE) accesses an entry in a list and optionally updates the corresponding dialog variables to the values in the list entry.
- "Get List Multiple Entries (QUIGETLM) API" on page 45 (QUIGETLM) accesses one or more entries in a list and updates the corresponding dialog variables with the values contained in the list entry.
- "Remove List Entry (QUIRMVLE) API" on page 50 (QUIMVLE) removes the list entry identified by the value of the current entry pointer for the list.
- "Retrieve List Attributes (QUIRTVLA) API" on page 52 (QUIRTVLA) retrieves list attributes
- "Set List Attributes (QUISETLA) API" on page 54 (QUISETLA) sets list attributes
- "Update List Entry (QUIUPDLE) API" on page 57 (QUIUPDLE) updates the list entry identified by the current entry pointer for the list or the extended action entry.

"User Interface Manager APIs," on page 1 | APIs by category

## Add List Entry (QUIADDLE) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Input | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | List name | Input | Char(10) |
| 6 | Option | Input | Char(4) |
| 7 | List entry handle | Output | Char(4) |
| 8 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Add List Entry (QUIADDLE) API adds one new entry to a list. The new entry is inserted immediately before or immediately after the entry identified by the current entry pointer for the list. The

new entry can also be inserted at the beginning or at the end of the list. On return to the application program, the current entry points to the newly inserted entry.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

A value assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
INPUT; CHAR(*)

Program storage containing values of one or more dialog variables, from which dialog variable values are copied. Dialog variables are copied in the order specified in the variable record definition.

If the variable record name parameter specifies the name of a variable record defined in the panel group for an open application, dialog variables are copied from the variable buffer to the application variable pool before the list entry is added. This parameter operation is the same as using the Put Dialog Variable (QUIPUTV) API immediately before the QUIADDLE API.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer. The buffer must be large enough to contain all the dialog variables in the variable record definition specified in the variable record name parameter. If the buffer is not large enough, an error condition is reported.

**Variable record name**
INPUT; CHAR(10)

The name of the variable record that determines which dialog variables are copied from the variable buffer to the application variable pool. The variable record must be defined in the panel group for the open application. The following special value can be used:

**NONE* The QUIPUTV API is not used during the QUIADDLE API. The variable buffer parameter and variable buffer length are ignored.

**List name**
INPUT; CHAR(10)

The name of the list to which an entry is added. If the list is not currently active in the open application, it is activated by this API.

**Option**
INPUT; CHAR(4)

The location of the new entry in the list. When an entry is added to the list, the current entry is always changed to point to the new list entry.

One of the following values must be specified to indicate the new entry's location:

*FRST* Added as the first entry in the list
*LAST* Added as the last entry in the list
*NEXT* Added after the current entry
*PREV* Added before the current entry

**List entry handle**
OUTPUT; CHAR(4)

A value representing an entry in a UIM list and returned to the application program representing the current entry in the list until it is removed from the list, even if other entries are inserted and removed from the list. This value is the handle of the entry just inserted in the list.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6AA1 E | The value of the action field is not correct at this time. Reason code &5. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A2B E | Value for Option parameter not valid. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A36 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A37 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF6A9D E | Size limit reached for list &4. |
| CPF6A90 E | Value not correct. Reason code &3. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A93 E | Operation not valid when current entry is &5. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Add List Multiple Entries (QUIADDLM) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Input | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | List name | Input | Char(10) |
| 6 | Option | Input | Char(4) |
| 7 | List entry handle | Output | Char(4) |
| 8 | Number of records | Input | Binary(4) |
| 9 | Record numbers | Input | Char(*) |
| 10 | Record size | Input | Binary(4) |
| 11 | Record count | Output | Binary(4) |
| 12 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Add List Multiple Entries (QUIADDLM) API adds one or more new entries to a list. The new entries are inserted immediately before or immediately after the entry identified by the current entry for the list. They can also be inserted at the beginning or end of the list. On return to the application program, the current entry points to the most recently inserted entry.

The contents of all dialog variables corresponding to columns in the list are saved in each added entry. When the operation completes successfully, the corresponding dialog variables contain the values from the last list entry successfully added.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
INPUT; CHAR(*)

The program buffer from which dialog variable values are copied. The dialog variables are copied in the order specified in the variable record definition.

If the variable record name parameter specifies the name of a variable record defined in the panel group for an open application, dialog variables are copied from the variable buffer to the application variable pool before the list entry is added. This parameter operation is the same as using the Put Dialog Variable (QUIPUTV) API immediately before the Add List Multiple Entry (QUIADDLM) API.

The variable buffer must be large enough to contain all the variables specified in the variable record definition.

When the number of records parameter has a value greater than one and the record numbers parameter has a value of zero, the size of the variable buffer must be at least equal to the value on the number of records parameter multiplied by the value on the record size parameter.

When the number of records parameter has a value greater than one and the record numbers parameter has a nonzero value, the size of the variable buffer must be at least equal to the largest value in the array of record numbers multiplied by the value of the record size parameter.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer. The buffer must be large enough to contain all the dialog variables in the variable record definition specified in the variable record name parameter.

**Variable record name**
INPUT; CHAR(10)

The name of the variable record determining which dialog variables are copied from the variable buffer to the application variable pool. The variable record must be defined in the panel group for the open application.

The following special value can be used:

**NONE* The QUIPUTV API is not used during the QUIADDLM API. The variable buffer, variable buffer length, number of records, record size, and record numbers parameters are ignored.

**List name**
INPUT; CHAR(10)

The name of the list to which entries are added. If the list is not currently active in the open application, it is activated by this API.

**Option**
INPUT; CHAR(4)

The location of the new entry in the list. When an entry is added to the list, the current entry is always changed to point to the new list entry.

One of the following values must be specified to indicate the new entry's location:

*FRST* Added as the first entry in the list
*LAST* Added as the last entry in the list
*NEXT* Added after the current entry
*PREV* Added before the current entry

**List entry handle**
OUTPUT; CHAR(4)

The list entry handle returned to the application program from the current entry. This value is the handle of the entry just inserted into the list. If more than one list entry is added, this parameter contains the list entry handle of the last entry successfully added.

**Number of records**
INPUT; BINARY(4)

The total number of entries the application program wants to add to the list. If the variable record name parameter specifies the name of a variable record defined in the panel group for this open application, this parameter can be used with the record size, record count, and record numbers parameters to add multiple entries to the list. When the variable record name parameter has the value *NONE, the number of records parameter is ignored.

The following special value can be used:

*1* Only one entry is added to the list. The record size, record count, and record numbers parameters are ignored.

When the number of records parameter is greater than 1, the variable buffer parameter must contain all the entries to be added to the list. The record size parameter defines the size of each record within the variable buffer, and is used to calculate the offset of each record from the first record. The variable record, identified by the variable record name parameter, defines the format of each record.

**Record numbers**
INPUT; CHAR (*)

An array of record numbers. Each entry in the array is a BINARY(4) value from 1 to 32767. The array specifies the order in which entries are added to the list. The first record number defines the first entry added to the list, the second record number defines the second entry, and so forth. Each record number specifies a record from the variable buffer containing the values for an entry to be added to the list. The dimension of the array must be at least as large as the value for the number of records parameter.

The following special value can be used in the first array element:

*0* Records from the variable buffer are used in sequential order. The array of record numbers needs to have only one element.

When the record numbers parameter is 0, the size of the variable buffer must be at least equal to the value specified on the number of records parameter multiplied by the value on the record size parameter. When the record numbers parameter has a nonzero value, the size of the variable buffer must be at least equal to the largest value specified in the record numbers array multiplied by the record size value.

The record numbers parameter allows applications to add more than one list entry from data structures already available to the application program. This is useful when the existing order of records in the data structures is not desired for the list entries, or when certain records should not be added to the list.

**Record size**
INPUT; BINARY(4)

The size of each record within the variable buffer when multiple records are added to the list. The record size is also used to calculate the offset of each record after the first record.

When the number of records parameter is 1, this parameter is ignored.

**Record count**
OUTPUT; BINARY(4)

The number of list entries actually added. Validation is done on packed or zoned data values in each record. If an error is found, the Add List Multiple Entries (QUIADDLM) API ends at that point with an exception; unusable data causes a partial update of the list. This parameter returns the number of list entries successfully added to the application program, even if an error occurs.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6AA1 E | The value of the action field is not correct at this time. Reason code &5. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A06 E | Record size or record number too large. |
| CPF6A2B E | Value for Option parameter not valid. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A30 E | Value for Record Numbers parameter not valid. |
| CPF6A36 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A37 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF6A9D E | Size limit reached for list &4. |
| CPF6A90 E | Value not correct. Reason code &3. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A93 E | Operation not valid when current entry is &5. |

| Message ID | Error Message Text |
|---|---|
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Delete List (QUIDLTL) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | List name | Input | Char(10) |
| 3 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Delete List (QUIDLTL) API deletes an active list and provides a way for the application to start over with a new list.

QUIDLTL does not need to be used before the Close Application (QUICLOA) API because all lists associated with an open application are automatically deleted when the application is closed.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or Open Print Application (QUIOPNPA) API when the application is opened.

**List name**
INPUT; CHAR(10)

The name of the list to be deleted. If the list is not currently active in the open application, an error is reported. A list is made active the first time an entry is inserted with the Add List Entry (QUIADDLE) API or Add List Multiple Entries (QUIADDLM) API, or when the list attributes are set with the Set List Attributes (QUISETLA) API.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |

| Message ID | Error Message Text |
|---|---|
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A92 E | List &4 not active. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

# Get List Entry (QUIGETLE) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Output | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | List name | Input | Char(10) |
| 6 | Positioning option | Input | Char(4) |
| 7 | Copy option | Input | Char(1) |
| 8 | Selection criteria | Input | Char(20) |
| 9 | Selection handle | Input | Char(4) |
| 10 | Extend option | Input | Char(1) |
| 11 | List entry handle | Output | Char(4) |
| 12 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Get List Entry (QUIGETLE) API accesses an entry in a list and optionally updates the corresponding dialog variables to the values in the list entry.

The entry accessed depends on the positioning option parameter and the value of the current entry pointer for the list. On return to the application program, the current entry pointer in the list points to the entry retrieved, except when retrieving the extended action entry. The extended action entry is a pseudo list entry that can be retrieved using this API, and can be updated using the Update List Entry (QUIUPDLE) API.

If requested, the corresponding dialog variables contain the values from the list entry retrieved.

If an error variable corresponding to a field in the list entry is specified in the variable record, identified by the variable record name parameter, it is set according to the error state of the corresponding dialog variable.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) or Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
OUTPUT; CHAR(*)

The program buffer into which dialog variable values are copied. The dialog variables are copied in the order specified in the variable record definition.

If the variable record name parameter specifies the name of a variable record defined in the panel group for this open application, dialog variables are copied into the variable buffer from the application variable pool after the list entry is retrieved. This parameter does the same job as using the Get Dialog Variable (QUIGETV) API immediately after the QUIGETLE API.

The variable buffer must be large enough to contain all the variables specified in the variable record definition.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer. The buffer must be large enough to contain all the dialog variables in the variable record definition, which is specified in the variable record name parameter.

**Variable record name**
INPUT; CHAR(10)

The name of the variable record that determines which dialog variables are copied between the application variable pool and the variable buffer. The variable record must be defined in the panel group for the open application. The following special value can be used:

**NONE* The QUIGETV API is not done during the QUIGETLE API; the parameter for the variable buffer is ignored when this value is used.

**List name**
INPUT; CHAR(10)

The name of the list from which an entry is retrieved. If the list is not currently active in the open application, an error is reported. A list is made active the first time an entry is inserted with the Add List Entry (QUIADDLE) API or Add List Multiple Entries (QUIADDLM) API, or when the list attributes are set with the Set List Attributes (QUISETLA) API.

**Positioning option**
INPUT; CHAR(4)

The placement of the current entry pointer to the list entry specified. If a positioning error occurs, the current list position is not changed. One of the following values must be specified to set the entry pointer to the appropriate position:

| *Value* | **Pointer Position** |
|---|---|
| *BOT* | The bottom of the list. A special position just after the last list entry in a list that is complete at the bottom. |
| *FRST* | The first entry in the currently built list. If the list is empty, an error is reported. |
| *FSLT* | The entire list is searched in a forward direction to locate an entry satisfying the condition specified by the selection criteria parameter. If the list is empty, an error is reported. The current entry pointer for the list entry is repositioned at the first entry in the list and the search proceeds forward, including the first entry. The search does not wrap. Note that if the list is incomplete at the top, the incomplete list processing program will not be called to complete the list at the top. It will start searching from the first entry in the currently built list. The incomplete list processing program will be called to fill in the entries at the bottom if the list is incomplete at the bottom. |
| *HNDL* | The list entry specified by the selection handle parameter. |

*LAST* The last entry in the currently built list. If the list is empty, an error is reported.

*LSLT* The entire list is searched in a backward direction to locate an entry satisfying the condition specified by the selection criteria parameter. If the list is empty, an error is reported. The current list entry pointer is repositioned to the last entry in the list and the search proceeds backward, including the last entry. The search does not wrap. Note that if the list is incomplete at the bottom, the incomplete list processing program will not be called to complete the list at the bottom. It will start searching from the last entry in the currently built list. The incomplete list processing program will be called to fill in the entries at the top if the list is incomplete at the top.

*NEXT* The current entry pointer is advanced one entry, pointing to the next entry in the list.

*NSLT* The list is searched in a forward direction to locate an entry satisfying the condition specified by the selection criteria parameter. The search starts with the entry after the current list entry pointer and does not wrap.

*PREV* The current entry pointer is moved back one entry, pointing to the previous entry in the list.

*PSLT* The list is searched in a backward direction to locate an entry satisfying the condition specified by the selection criteria parameter. The search starts with the entry before the current list entry pointer and does not wrap.

*SAME* The current entry pointer remains unchanged, pointing to the same entry in the list.

*TOP* The top of the list. A special position just prior to the first list entry in a list that is complete at the top.

**Note:** After running the QUIGETLE API once with the positioning option parameter of FSLT or LSLT, the application program should change the positioning option parameter to NSLT or PSLT. It does this to avoid repeatedly positioning at the beginning or end of the list and searching the same list entries.

**Copy option**

INPUT; CHAR(1)

Determines whether or not the data values in the list entry are copied into the corresponding dialog variables when positioning is complete. One of the following values must be specified:

*Y* The data values in the current list entry after the positioning operations are performed are copied into corresponding dialog variables. This value must be specified if the variable record name parameter specifies the name of a variable record defined in the panel group for the open application. This value is not allowed when the current entry is positioned to TOP or BOT.

*N* The data values in the current list entry after positioning the list are not copied into corresponding dialog variables.

**Selection criteria**

INPUT; CHAR(20)

The selection criteria used when positioning the list by variable selection. Positioning by variable selection allows the application program to search forward to find the next entry, or backward to find a previous entry satisfying a given condition. The search is not dependent on sorted list entries, but the application program might need to build the list in sorted order to get a consistent result when application comparison operators are other than equal (EQ) or not equal (NE).

This parameter is only used when one of the following values is specified for the positioning option parameter: FSLT, LSLT, NSLT, or PSLT. It is ignored if any other value is specified.

The relational condition defined by this parameter must exist between the value of a dialog variable and that dialog variable's corresponding value in a list entry when selection positioning is performed.

The first 10 characters of this parameter must contain a comparison operator. The operator must be left-adjusted and padded with blanks.

One of the following values must be specified for the comparison operator:

*EQ* The list entry value equals the dialog variable value.

*NE* The list entry value is not equal to the dialog variable value.
*GT* The list entry value is greater than the dialog variable value.
*LT* The list entry value is less than the dialog variable value.
*GE* The list entry value is greater than or equal to the dialog variable value.
*LE* The list entry value is less than or equal to the dialog variable value.

The second 10 characters of this parameter must contain the name of the dialog variable used in the comparison. The dialog variable name specified must be defined in the list being searched.

**Selection handle**
INPUT; CHAR(4)

The list entry handle used when positioning the current entry pointer to a specific entry. This parameter is used only when the positioning option parameter has the value HNDL; it is ignored if any other value is specified.

The following special value can be specified:

*EXTE* Retrieves the contents of the extended action entry. The contents of the extended action entry are copied to the associated dialog variables in the variable pool. However, the current entry pointer for the list is not changed when the extended action entry is retrieved.

A list entry handle uniquely distinguishes an entry until it is removed from the list, even if other entries are inserted and removed from the list.

**Extend option**
INPUT; CHAR(1)

Indicates whether or not an incomplete list is automatically extended in the attempt to retrieve the requested list entry. The list can be extended when one of the following values is specified for the positioning option parameter: NEXT, PREV, TOP, BOT, NSLT, PSLT, FSLT, or LSLT. This parameter is ignored and the list is not extended if any other value is specified for the positioning option parameter.

One of the following values must be specified:

*Y* The list is extended, if necessary, to find the requested list entry.
*N* The list is not extended to find the requested list entry. This option can be used if the program calling the QUIGETLE API needs to process only entries already in the list.

**List entry handle**
OUTPUT; CHAR(4)

The list entry handle from the current entry pointer in the list. This value is the handle of the entry to which the list was last positioned by this API. The following special values may be returned:

*TOP* The current entry pointer is positioned at the top of the list.
*BOT* The current entry pointer is set at the bottom of the list.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A14 E | Program defined by variable &4 cannot be called. |
| CPF6A15 E | Errors occurred in list exit program. |
| CPF6A2C E | Value for Option parameter not valid. |
| CPF6A2D E | Value for Selection Criteria parameter not valid. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A27 E | Value for Extend Option parameter not valid. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A92 E | List &4 not active. |
| CPF6A93 E | Operation not valid when current entry is &5. |
| CPF6A95 E | List &4 either not complete or not extended. |
| CPF6A96 E | Variable &5 is not in list definition. |
| CPF6A97 E | Variable &5 not valid type for select comparison. |
| CPF6A98 E | Entry not found in list &4 in panel group &1 in &2. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V3R6

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

## Get List Multiple Entries (QUIGETLM) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Output | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | List name | Input | Char(10) |
| 6 | Positioning option | Input | Char(4) |
| 7 | Copy option | Input | Char(1) |
| 8 | Selection criteria | Input | Char(20) |
| 9 | Selection handle | Input | Char(4) |
| 10 | Extend option | Input | Char(1) |
| 11 | List entry handle | Output | Char(4) |
| 12 | Number of records | Input | Binary(4) |
| 13 | Record size | Input | Binary(4) |
| 14 | Record count | Output | Binary(4) |
| 15 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Get List Multiple Entries (QUIGETLM) API accesses one or more entries in a list and updates the corresponding dialog variables with the values contained in the list entry.

The entry accessed depends on the positioning option parameter and the value of the current entry pointer for the list. On return to the application program, the current entry pointer in the list points to the last entry retrieved, except when retrieving the extended action entry. The extended action entry can be retrieved using this API, and can be updated using the Update List Entry (QUIUPDLE) API.

When the operation completes successfully, the corresponding dialog variables contain the values from the last list entry retrieved.

If an error variable corresponding to a field in the list entry is specified in the variable record, which is identified by the variable record name parameter, it is set according to the error state of the corresponding dialog variable.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
OUTPUT; CHAR(*)

The program buffer into which dialog variable values are copied. The dialog variables are copied in the order specified in the variable record definition.

Dialog variables are copied into the variable buffer from the application variable pool after the list entry is retrieved. This parameter does the same job as using the Get Dialog Variable (QUIGETV) API immediately after the list entry is retrieved.

The variable buffer must be large enough to contain all the variables specified in the variable record definition.

When the number of records parameter is greater than 1, the size of the variable buffer must be at least equal to the value specified on the number of records parameter multiplied by the value specified on the record size parameter.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer. The buffer must be large enough to contain all the dialog variables in the variable record definition specified in the variable record name parameter.

**Variable record name**
INPUT; CHAR(10)

The name of the variable record that determines which dialog variables are copied between the application variable pool and the variable buffer. The variable record must be defined in the panel group for the open application.

The special value of *NONE may not be used with the QUIGETLM API.

**List name**
INPUT; CHAR(10)

The name of the list from which an entry is retrieved. If the list is not currently active in the open application, an error is reported. A list is made active the first time an entry is inserted with the

Add List Entry (QUIADDLE) or Add List Multiple Entries (QUIADDLM) API, or when the list attributes are set with the Set List Attributes (QUISETLA) API.

**Positioning option**
INPUT; CHAR(4)

Sets the current entry pointer to the list entry specified. If a positioning error occurs, the current list position is not changed. One of the following values must be specified to set the entry pointer to the appropriate position:

| *Value* | **Pointer Position** |
|---|---|
| *BOT* | The bottom of the list. A special position just after the last list entry in a list that is complete at the bottom. |
| *FRST* | The first entry in the currently built list. If the list is empty, an error is reported. |
| *FSLT* | The entire list is searched in a forward direction to locate an entry satisfying the condition specified by the selection criteria parameter. If the list is empty, an error is reported. The current pointer for the list entry is repositioned at the first entry in the list, and the search proceeds forward, including the first entry. The search does not wrap. |
| *HNDL* | The list entry specified by the selection handle parameter. |
| *LAST* | The last entry in the currently built list. If the list is empty, an error is reported. |
| *LSLT* | The entire list is searched in a backward direction to locate an entry satisfying the condition specified by the selection criteria parameter. If the list is empty, an error is reported. The current list entry pointer is repositioned to the last entry in the list, and the search proceeds backward, including the last entry. The search does not wrap. |
| *NEXT* | The current entry pointer is advanced one entry, pointing to the next entry in the list. |
| *NSLT* | The list is searched in a forward direction to locate an entry satisfying the condition specified by the selection criteria parameter. The search starts with the entry after the current list entry pointer, and does not wrap. |
| *PREV* | The current entry pointer is moved back one entry, pointing to the previous entry in the list. |
| *PSLT* | The list is searched in a backward direction to locate an entry satisfying the condition specified by the selection criteria parameter. The search starts with the entry before the current list entry pointer and does not wrap. |
| *SAME* | The current entry pointer remains unchanged, pointing to the same entry in the list. |
| *TOP* | The top of the list. A special position just prior to the first list entry in a list that is complete at the top. |

**Note:** After running the QUIGETLM API once with the positioning option parameter of FSLT or LSLT, the application program should change the positioning option parameter to NSLT or PSLT. It does this to avoid repeatedly positioning at the beginning or end of the list and searching the same list entries.

**Copy option**
INPUT; CHAR(1)

Determines whether or not the data values in the list entry are copied into the corresponding dialog variables when positioning is complete. One of the following values must be specified:

*Y* The data values in the current list entry after the positioning operations are performed are copied into corresponding dialog variables. This value must be specified if the variable record name parameter specifies the name of a variable record defined in the panel group for the open application. Y must be used if the number of records parameter is greater than 1. This value is not allowed when the current entry is positioned to TOP or BOT.

*N* The data values in the current list entry after positioning the list are not copied into corresponding dialog variables.

**Selection criteria**
INPUT; CHAR(20)

The selection criteria used when positioning the list by variable selection. Positioning by variable selection allows the application program to search forward to find the next entry, or backward to find a previous entry satisfying a given condition. The search is not dependent on sorted list

entries, but the application program might need to build the list in sorted order to get a consistent result when using comparison operators other than equal (EQ) or not equal (NE).

This parameter is only used when FSLT, LSLT, NSLT, or PSLT is specified for the positioning option parameter. It is ignored if any other value is specified.

The relational condition defined by this parameter must exist between the dialog variable value and that dialog variable's corresponding value in a list entry when selection positioning is performed.

The first 10 characters of this parameter must contain a comparison operator. The operator must be left-adjusted and padded with blanks.

One of the following values must be specified for the comparison operator:

*EQ* The list entry value equals the dialog variable value.
*NE* The list entry value is not equal to the dialog variable value.
*GT* The list entry value is greater than the dialog variable value.
*LT* The list entry value is less than the dialog variable value.
*GE* The list entry value is greater than or equal to the dialog variable value.
*LE* The list entry value is less than or equal to the dialog variable value.

The second 10 characters of this parameter contain the name of the dialog variable used in the comparison. The dialog variable name specified must be defined in the list being searched.

**Selection handle**
INPUT; CHAR(4)

The list entry handle used when positioning the current entry pointer to a specific entry. This parameter is used only when the positioning option parameter has the value HNDL; it is ignored if any other value is specified.

The following special value can be specified:

*EXTE* Retrieves the content of the extended action entry. The contents of the extended action entry are copied to the associated dialog variables in the variable pool. However, the current entry pointer for the list is not changed when the extended action entry is retrieved.

A list entry handle uniquely distinguishes an entry until it is removed from the list, even if other entries are inserted and removed from the list.

**Extend option**
INPUT; CHAR(1)

Indicates whether or not an incomplete list is automatically extended in the attempt to retrieve the requested list entry. The list can be extended when one of the following values is specified for the positioning option parameter: NEXT, PREV, TOP, BOT, NSLT, PSLT, FSLT, or LSLT. If any other value is specified for the positioning option parameter, this parameter is ignored and the list is not extended.

One of the following values must be specified:

*Y* The list is extended, if necessary, to find the requested list entry.
*N* The list is not extended to find the requested list entry. This option can be used if the program calling the QUIGETLE API needs to process only entries already in the list.

**List entry handle**
OUTPUT; CHAR(4)

The list entry handle from the current entry pointer in the list. This value is the handle of the entry to which the list was last positioned by this API.

The following special values may be returned.

*TOP* The current entry pointer is positioned at the top of the list.
*BOT* The current entry pointer is set at the bottom of the list.

**Number of records**
INPUT; BINARY(4)

The total number of entries to retrieve from the list. The following special value can be used:

*1* Only one entry is retrieved from the list. The record size and record count parameters are ignored when this value is used.

When this parameter is greater than 1, the variable buffer must contain space for all the entries retrieved from the list.

When this parameter is greater than 1 (multiple entries are being retrieved), the positioning option parameter must have one of the following values: NEXT, PREV, NSLT, PSLT, FSLT, or LSLT.

**Record size**
INPUT; BINARY(4)

The size of each record within the variable buffer when multiple records are retrieved from the list. This parameter also calculates the offset of each record after the first one.

When the number of records parameter is 1, this parameter is ignored.

**Record count**
OUTPUT; BINARY(4)

The number of list entries actually retrieved. When the number of records parameter is 1, this parameter is ignored.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| | |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A06 E | Record size or record number too large. |
| CPF6A13 E | Application &3 closed prematurely. |
| CPF6A14 E | Program defined by variable &4 cannot be called. |
| CPF6A15 E | Errors occurred in list exit program. |
| CPF6A2C E | Value for Option parameter not valid. |
| CPF6A2D E | Value for Selection Criteria parameter not valid. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A27 E | Value for Extend Option parameter not valid. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF6A90 E | Value not correct. Reason code &3. |
| CPF6A91 E | List &4 does not exist. |

| | |
|---|---|
| CPF6A92 E | List &4 not active. |
| CPF6A93 E | Operation not valid when current entry is &5. |
| CPF6A95 E | List &4 either not complete or not extended. |
| CPF6A96 E | Variable &5 is not in list definition. |
| CPF6A97 E | Variable &5 not valid type for select comparison. |
| CPF6A98 E | Entry not found in list &4 in panel group &1 in &2. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Remove List Entry (QUIRMVLE) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | List name | Input | Char(10) |
| 3 | Extend option | Input | Char(1) |
| 4 | List entry handle | Output | Char(4) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Remove List Entry (QUIRMVLE) API removes the list entry identified by the value of the current entry pointer for the list. The current entry pointer is always updated to the entry before the one removed. If the first list entry is removed, the current entry pointer is set to the top of the list if the list is complete at the top.

If the list is incomplete at the top, the UIM calls the incomplete list extension program to add another entry to the list.

If the list entry identified by the display position attribute parameter of the Set List Attributes (QUISETLA) API is removed, the display position attribute is set at the entry before the one that was removed. If the new display position attribute is the first entry in the list, the display position attribute is set at the top of the list (logically the entry before the first entry in the list) if the list is complete at the top.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**List name**
INPUT; CHAR(10)

The name of the list from which an entry is removed. If the list is not currently active in the open application, an error message is reported. A list is made active the first time an entry is inserted with the Add List Entry (QUIADDLE) API, Add List Multiple Entries (QUIADDLM) API, or its attributes are set with the QUISETLA API.

**Extend option**
INPUT; CHAR(1)

Specifies whether or not an incomplete list is automatically extended in an attempt to remove the first entry from a list that is incomplete at the top. One of the following values must be specified:

*Y* The list is extended, if necessary, to add at least one entry to the top of the list or to mark the list as complete at the top.
*N* The list is not extended to find the entry before the entry being removed, and the entry is not removed from the list.

**List entry handle**
OUTPUT; CHAR(4)

The list entry handle value from the current entry pointer. A list entry handle uniquely distinguishes an entry until it is removed from the list, even if other entries are inserted and removed from the list. A value of TOP indicates that the current entry pointer is positioned at the top of the list.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A13 E | Application &3 closed prematurely. |
| CPF6A14 E | Program defined by variable &4 cannot be called. |
| CPF6A15 E | Errors occurred in list exit program. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A27 E | Value for Extend Option parameter not valid. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A92 E | List &4 not active. |
| CPF6A93 E | Operation not valid when current entry is &5. |
| CPF6A94 E | Incomplete list &4 requires extension. |
| CPF6A95 E | List &4 either not complete or not extended. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

# Retrieve List Attributes (QUIRTVLA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | List name | Input | Char(10) |
| 3 | Receiver | Output | Char(*) |
| 4 | Receiver length | Input | Binary(4) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Retrieve List Attributes (QUIRTVLA) API retrieves the following list attributes:

- The list contents attribute, indicating whether or not all entries are present in the list, and which entries are missing if it is incomplete
- The name of the dialog variable that identifies the program called when the UIM needs to add entries to an incomplete list
- The display position attribute, which is the list entry handle for the entry presented at the top-most row of any list area that displays a list
- The allow trim attribute, which indicates whether or not the UIM trims a full list when adding new entries

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or by the Open Print Application (QUIOPNPA) API when the application is opened.

**List name**
INPUT; CHAR(10)

The name of the list whose attributes are retrieved. If the list is not currently active in the open application, an error is reported. A list is made active the first time an entry is inserted with the Add List Entry (QUIADDLE) or Add List Multiple Entries (QUIADDLM) API, or the fist time the list's attributes are set with the Set List Attributes (QUISETLA) API.

**Receiver**
OUTPUT; CHAR(*)

The current attributes of the list. For the format of the receiver variable, see "Format of Data Returned" on page 53.

**Receiver length**
INPUT; BINARY(4)

The amount of data the application program is prepared to receive. If the length specified is larger than the amount of data available, the receiver is not changed beyond the amount of data available.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Format of Data Returned

The format of the data available, returned in the receiver parameter, is as follows:

*CHAR(4)* The list contents attribute, indicating whether or not the list contains all or some of the entries available for display or printing. For more information about the meaning of each possible return value, see "Set List Attributes (QUISETLA) API" on page 54. The following values can be returned in this parameter:

*ALL* The list is complete. This value is returned when either the QUISETLA API has not been called to set the list attribute or when ALL is the last list contents attribute specified on a call to the QUISETLA API for the list.

*TOP* Only the top part of an incomplete list is available. This value is returned when TOP is the last list contents attribute specified on a call to the QUISETLA API for the list.

*BOT* Only the bottom part of an incomplete list is available. This value is returned when BOT is the last list contents attribute specified on a call to the QUISETLA API for the list.

*MORE* Only the middle part of an incomplete list is available. This value is returned when MORE is the last list contents attribute specified on a call to the QUISETLA API for the list.

*CHAR(10)* The name of the dialog variable identifying the program the UIM calls when more entries are needed in an incomplete list.

*CHAR(4)* The display position attribute, which is the list entry handle at the top of the list area on the next panel that displays this list. The UIM does not use the display position attribute for print applications.

The value returned is the handle for a specific entry in the list, or one of the following special values:

*TOP* The top entries in the list are displayed.

*BOT* The bottom entries in the list are displayed.

If the list entry identified by the display position attribute of a list is removed before the list is displayed on a panel, the UIM adjusts the attribute to display the entry before the one that was removed. The value returned by this API is not meaningful if entries are removed from the list after the display position attribute is retrieved.

*CHAR(1)* The allow trim attribute, which indicates whether or not the UIM trims the list when a new list entry causes the list to exceed its maximum size. For additional details, see "Set List Attributes (QUISETLA) API" on page 54.

One of the following values is returned:

*Y* The UIM automatically trims the list.

*N* The UIM does not automatically trim the list.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A92 E | List &4 not active. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

# Set List Attributes (QUISETLA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | List name | Input | Char(10) |
| 3 | List contents | Input | Char(4) |
| 4 | Program dialog variable | Input | Char(10) |
| 5 | Display position attribute | Input | Char(4) |
| 6 | Allow trim attribute | Input | Char(1) |
| 7 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Set List Attributes (QUISETLA) API sets the following list attributes:

- The list contents attribute, which indicates whether or not all entries are present in the list and which entries are missing if it is incomplete
- The name of the dialog variable that identifies the program the UIM calls when additional entries are needed in an incomplete list
- The display position attribute, which is the list entry handle for the entry presented at the top row of a list area displaying the list
- An attribute indicating whether or not the UIM trims the list if the maximum list size is exceeded when adding a new entry to the list

If the QUISETLA API has not been called for a list since the first entry was inserted, the list contents default value is ALL and the list is complete. An error message is displayed if the user attempts to page beyond either end of the list. The UIM does not call a program for more entries or return to the program that called the Display Panel (QUIDSPP) API.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or by the Open Print Application (QUIOPNPA) API when the application is opened.

**List name**
INPUT; CHAR(10)

The name of the list whose attributes are changed. If the list is not currently active in the open application, it is activated by this API.

**List contents**
INPUT; CHAR(4)

Indicates whether or not the list contains all or some of the entries available. One of the following values must be specified:

*SAME* This list contents attribute of the list is not changed.

*ALL* The entire list has been built. A message is displayed if the user attempts to page beyond the top or the bottom of the list.

*TOP* Only the top part of the list is built. A message is displayed if the user attempts to page beyond the beginning of the list. If the user attempts to page beyond the bottom of the list, the program specified by the program dialog variable parameter is called by the UIM, with parameters indicating the need for more entries at the bottom of the list.

*BOT* Only the bottom part of the list is built. A message is displayed if the user attempts to page beyond the bottom of the list. If the user attempts to page beyond the top of the list, the program specified by the program dialog variable parameter is called by the UIM, with parameters indicating the need for more entries at the top of the list.

*MORE* Only the middle of the list is built. If the user attempts to page beyond either the top or the bottom the list, the program specified by the program dialog variable parameter is called by the UIM, with parameters indicating the need for more entries at the top or bottom of the list.

If the application program marks the list complete in a given direction, the application program cannot subsequently mark the list incomplete in that same direction.

**Program dialog variable**
INPUT; CHAR(10)

The name of a dialog variable in an open application. This dialog variable contains information identifying the program called by the UIM when more entries must be added to an incomplete list. The dialog variable must be defined and set properly according to the rules used for the CALL dialog command.

For a description of the CALL dialog command, see the Application Display Programming manual. For a description of the interface between the UIM and the incomplete list exit program, see "Using the User Interface Manager Exit Programs" on page 100.

The following special value can be used:

**SAME* The program dialog variable is not changed.

**Display position attribute**
INPUT; CHAR(4)

The list entry that should be positioned at the top of the list area on the next panel that displays this list. The UIM does not use the display position attribute when printing the list.

The value specified must be the list entry handle currently existing in the list or one of the following special values:

*SAME* The display position attribute of the list is not changed.

*TOP* The entries at the top of the list should be positioned at the top of the list area on the next panel.

*BOT* The bottom *n* entries of the list should be displayed in the list area on the next panel, where *n* is the number of list entries that can be displayed in the current view of the list. If the list contains fewer than *n* list entries, the complete list is displayed.

If the list entry identified by the display position attribute of a list is removed before the list is displayed on a panel, the UIM automatically adjusts the attribute to display the entry before the one removed. If there is no previous entry in the list, the attribute is set at the top of the list, and the next panel displaying the list shows the first entry in the list.

During incomplete list extension, the application program must specify SAME for the display position attribute.

**Allow trim attribute**
INPUT; CHAR(1)

Indicates whether or not the UIM should trim the list during the Add List Entry (QUIADDLE) or the Add List Multiple Entries (QUIADDLM) API if adding the new entry exceeds the maximum list size. The UIM trims only lists that have been marked as incomplete by using the QUISETLA API. When the UIM removes the list entry, the UIM marks the list incomplete in the direction from which the entry was removed, regardless of whether or not the list was previously incomplete in this direction.

If an application is adding an entry to the bottom of the list when the maximum size list is reached, the UIM removes the first list entry and marks the list incomplete at the top. The incomplete list exit program of the application must be prepared to add list entries in either direction of an incomplete list.

When the incomplete list exit program is called to add list entries, the QUIADDLE or QUIADDLM API can be called to add entries in the middle of the list. If the entries are added to the middle of the list when the list is complete; trimming does not take place and an error is reported indicating that the maximum list size has been reached.

One of the following values must be specified:

*S* The allow trim attribute of the list should not be changed.
*Y* The UIM automatically trims the list.
*N* The UIM does not automatically trim the list.

**Error code**

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA0 E | Request is not allowed when extending a list that is not complete. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A9A E | Dialog variable required to specify incomplete list. |
| CPF6A9B E | Dialog variable &5 not the correct type. |
| CPF6A9C E | List entry handle not correct. |
| CPF6A9E E | Value for Allow Trim parameter not valid. |
| CPF6A9F E | Attribute for list &4 not valid. |
| CPF6A90 E | Value not correct. Reason code &3. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A99 E | Dialog variable &5 not found in panel group &1 in &2. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

# Update List Entry (QUIUPDLE) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Input | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | List name | Input | Char(10) |
| 6 | Option | Input | Char(4) |
| 7 | List entry handle | Output | Char(4) |
| 8 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Update List Entry (QUIUPDLE) API updates the list entry identified by the current entry pointer for the list or the extended action entry. The current contents of all dialog variables corresponding to dialog variables in the list are saved in the entry. The current entry pointer of the list is not changed by this operation.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
INPUT; CHAR(*)

The program buffer from which dialog variable values are copied. The dialog variables are copied in the order specified in the variable record definition.

If the variable record name parameter specifies the name of a variable record, which is defined in the panel group for this open application, dialog variables are copied from the variable buffer to the application variable pool before the list entry is updated. The operation of this parameter is the same as using the Put Dialog Variable (QUIPUTV) API immediately before the QUIUPDLE API.

The variable buffer must be large enough to contain all the variables specified in the variable record definition.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer provided. The buffer must be large enough to contain all the dialog variables in the definition of the variable record, specified in the variable record name parameter.

**Variable record name**
INPUT; CHAR(10)

The name of the variable record that determines which dialog variables are copied between the variable buffer and the application variable pool. The variable record must be defined in the panel group for the open application.

The following special value can be used:

**NONE* The QUIPUTV API is not done during the QUIUPDLE API. The variable buffer parameter is ignored when this value is used.

**List name**

INPUT; CHAR(10)

The name of the list in which an entry is updated. If the list is not currently active in the open application, an error is reported. A list is made active the first time an entry is inserted with the Add List Entry (QUIADDLE) API or the Add List Multiple Entries (QUIADDLM) API, or its attributes are set with the Set List Attributes (QUISETLA) API.

**Option**

INPUT; CHAR(4)

The updated list entry. One of the following values must be specified:

*EXTE* The extended action entry is updated.
*SAME* The list entry identified by the current entry pointer is updated.

**List entry handle**

OUTPUT; CHAR(4)

The list entry handle for the updated list entry. A list entry handle uniquely distinguishes an entry until it is removed from the list, even if other entries are inserted and removed from the list.

When the option parameter has the value EXTE, this parameter returns the value EXTE, indicating that the extended action entry is updated.

**Error code**

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6AA1 E | The value of the action field is not correct at this time. Reason code &5. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A28 E | Value for Option parameter not valid. |
| CPF6A36 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A37 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF6A90 E | Value not correct. Reason code &3. |
| CPF6A91 E | List &4 does not exist. |
| CPF6A92 E | List &4 not active. |
| CPF6A93 E | Operation not valid when current entry is &5. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

## UIM Print APIs

The UIM print APIs are:

- "Add Print Application (QUIADDPA) API" (QUIADDPA) enables print functions in a previously opened display application by opening the printer file for the application.
- "Print Help (QUHPRTH) API" on page 62 (QUHPRTH) prints help information from help modules named when this API is called.
- "Print Panel (QUIPRTP) API" on page 63 (QUIPRTP) prints a panel to the printer file for an opened print application.
- "Remove Print Application (QUIRMVPA) API" on page 65 (QUIRMVPA) stops print functions in a previously opened display application.

"User Interface Manager APIs," on page 1 | APIs by category

## Add Print Application (QUIADDPA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Qualified printer file name | Input | Char(20) |
| 3 | Alternative file name | Input | Char(10) |
| 4 | Share open data path | Input | Char(1) |
| 5 | User data | Input | Char(10) |
| 6 | Error code | I/O | Char(*) |

Optional Parameter Group:

| | | | |
|---|---|---|---|
| 7 | Open data receiver | Output | Char(*) |
| 8 | Length of open data receiver | Input | Binary(4) |
| 9 | Length of available open data | Output | Binary(4) |

Default Public Authority: *USE
Threadsafe: No

The Add Print Application (QUIADDPA) API enables print functions in a previously opened display application by opening the printer file for the application. The QUIADDPA API and the Remove Print Application (QUIRMVPA) API are used in pairs to add and remove printing from applications.

Because the QUIADDPA API requires an open application for display, this print function does not work in a batch environment. For printing in batch, use the Open Print Application (QUIOPNPA) API.

## Authorities and Locks

*Library Authority*
*READ

*Printer Device File Authority*
*USE

*Printer Device File Lock*
*SHRNUP

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API when the application is opened.

**Qualified printer file name**
INPUT; CHAR(20)

The name of the printer device file used for print operations. The first 10 characters contain the name of the *FILE object, and the second 10 characters contain the name of the library in which the printer device file resides. These special values can be used for the library name:

**CURLIB* The job's current library
**LIBL* The library list

The user must have *USE authority to the file named by this parameter.

**Alternative file name**
INPUT; CHAR(10)

An alternative name for the spooled file. The following special value can be used:

**NONE* There is no alternative name for the spooled file. The name of the spooled file is the name of the printer device file.

**Share open data path**
INPUT; CHAR(1)

Indicates whether or not the open data path (ODP) for the printer file is shared. Sharing the ODP allows multiple UIM applications to print to the same spooled file. One of the following values must be used:

*Y* The ODP is shared.
*N* The ODP is not shared.
*F* Use the share value of the printer file.

**User data**
INPUT; CHAR(10)

User data associated with the spooled file. This data becomes an attribute of the spooled file. The following special values can be used:

**FILE* The user data of the spooled file will be set to the user data value of the printer file being opened.
**NONE* There is no user data associated with the spooled file. The user data value of the printer file being opened will be set to blanks.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Optional Parameter Group

**Open data receiver**
OUTPUT; CHAR(*)

The variable that is to receive the open data information requested. For the format of the open data receiver variable, see "Format of Data Returned."

**Length of open data receiver**
INPUT; BINARY(4)

The amount of data the application program is prepared to receive. If the length specified is larger than the amount of data available, the receiver is not changed beyond the amount of data available. If the length specified is larger than the actual length of the open data receiver parameter, unpredictable results may occur.

**Length of available open data**
OUTPUT; BINARY(4)

The length of all open data available. All available open data is returned if enough space is provided.

## Format of Data Returned

The format of the data available, returned in the open data receiver parameter, is as follows:

**CHAR(1)**
Whether or not an error occurred while attempting to obtain the conversion tables needed to process the panel group. The conversion tables are needed when the CHRID attribute of the panel group is not equal to the CHRID attribute of the device, or when the CHRID attribute of the panel group is *JOBCCSID and the job CCSID is not equal to the device CHRID. A CPD6A2A diagnostic message will be logged in the job log for each conversion table that is not found.

One of the following values is returned:

*N* No error occurred while obtaining the conversion tables or the conversion tables were not necessary.
*Y* An error occurred while obtaining the conversion tables.

**CHAR(1)**
Whether or not the conversion of data from the job to the device and from the device to the job will result in loss of fidelity of the data. Conversion will be done when the CHRID attribute of the panel group is not equal to the CHRID attribute of the device, or when the CHRID attribute of the panel group is *JOBCCSID and the job CCSID is not equal to the device CHRID.

One of the following values is returned:

*N* No loss of fidelity will occur.
*Y* Loss of fidelity may occur on the conversions.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A1A E | Application already has an open print file. |
| CPF6A1C E | Unable to add print function. |
| CPF6A1E E | Object cannot be used with this device or print file. |
| CPF6A11 E | Value is not correct. Reason code is &3. |
| CPF6A20 E | Print code page not identical to display code page. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A3A E | Value for Open Data Receiver is not valid. Reason code &1 |

| Message ID | Error Message Text |
|---|---|
| CPF9850 E | Override of printer file &1 not allowed. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Print Help (QUHPRTH) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Help identifier array | Input | Char(*) |
| 2 | Number of help identifiers | Input | Binary(4) |
| 3 | Help title | Input | CHAR(55) |
| 4 | Error code | I/O | Char(*) |

Threadsafe: No

The Print Help (QUHPRTH) API prints help information from help modules named when this API is called.

## Warning: Temporary Level 3 Header

### Restriction

This API is only valid in an interactive job. It cannot be run from a batch process.

## Authorities and Locks

*Library Authority*
*READ

*Panel Group Authority*
*USE

*Panel Group Lock*
*SHRRD

## Required Parameter Group

**Help identifier array**
INPUT; CHAR(*)

An array of the help identifiers to print. The list can contain up to 2000 items. Each item has two parts:

**Qualified help panel group name**
CHAR(20)

The panel group (*PNLGRP) object that contains the help module to be printed, and the library in which it is located. (A **panel group** is an object with an object type of *PNLGRP. It contains display panels, print panels, or help modules.)

The first 10 characters contain the panel group object name, and the second 10 characters contain the library name. You can use these special values for the library name:

| | |
|---|---|
| **CURLIB* | The job's current library |
| **LIBL* | The library list |

**Help module name**
CHAR(32)

The name specified on the NAME attribute of a :HELP. tag in the panel group object. The name must be specified using uppercase, alphabetic characters.

**Number of help identifiers**
INPUT; BINARY(4)

The number of help identifiers in the help identifier array parameter. The number must be between 1 and 2000.

**Help title**
INPUT; CHAR(55)

The title to print if no title is found in the help panel group object.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6Exx E | All CPF6Exx messages could be signalled. xx is from 01 to FF. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V3R1

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Print Panel (QUIPRTP) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Print panel name | Input | Char(10) |
| 3 | Eject option | Input | Char(1) |
| 4 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Print Panel (QUIPRTP) API prints a panel to the printer file for an opened print application. The values for all output fields used in the panel definition are taken from dialog variables in the variable pool. If the panel contains list areas, the values are also taken from list entries in the lists associated with the open application.

If the panel contains a list area that is incomplete at the bottom, the UIM automatically calls the program identified by the program dialog variable parameter of the Set List Attributes (QUISETLA) API to acquire more list entries. The program is called repeatedly until either the requested number of entries is added to the list or the list is marked complete at the bottom. For lists that are incomplete at the top, printing begins with the first entry in the list.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**Print panel name**
INPUT; CHAR(10)

The name of the print head panel or print panel defined in the panel group for the open application.

**Eject option**
INPUT; CHAR(1)

Determines whether or not this panel begins on a new page. An automatic page eject is done when a print head panel is printed after a print panel. However, even if Y is specified on the QUIPRTP API when the next print panel is printed, it does not cause a second page eject.

One of the following values must be used:

*Y* The panel is printed at the top of a new page.
*N* The panel is not always printed at the top of a new page.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A1F E | An active display already exists for this application. |
| CPF6A11 E | Value is not correct. Reason code is &3. |
| CPF6A13 E | Application &3 closed prematurely. |
| CPF6A14 E | Program defined by variable &4 cannot be called. |
| CPF6A15 E | Errors occurred in list exit program. |
| CPF6A18 E | Print heading must be specified first. |
| CPF6A19 E | Prologue is only allowed in first heading. |
| CPF6A23 E | Page length too small to print the list column headings. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A3B E | Application not open for print. |
| CPF6A3E E | Application not open for display. |
| CPF6A3F E | Panel &4 was not found in panel group &1. |
| CPF6A50 E | Error was found during display file or printer file operation. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2



# Remove Print Application (QUIRMVPA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Close option | Input | Char(1) |
| 3 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Remove Print Application (QUIRMVPA) API stops print functions in a previously opened display application. The Add Print Application (QUIADDPA) API and the QUIRMVPA API are used in pairs to add and remove printing from display applications.

The QUIRMVPA API closes the printer file for the application. If the QUIRMVPA API is not performed before the application is closed, the Close Application (QUICLOA) API performs the QUIRMVPA API.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API when the application is opened.

**Close option**
INPUT; CHAR(1)

Specifies whether or not to perform a normal or abnormal close operation on the printer file. One of the following values must be specified:

*A* An abnormal close operation is performed, and the trailer message is not printed.
*M* A normal close operation is performed.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A1B E | Application does not have an open print file. |
| CPF6A11 E | Value is not correct. Reason code is &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |

| Message ID | Error Message Text |
|---|---|
| CPF6A25 E | Return code length of &1 not valid. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## UIM Variable Pool APIs

The UIM variable pool APIs are:

- "Get Dialog Variable (QUIGETV) API" (QUIGETV) allows a program to obtain the value of one or more dialog variables by specifying the application program's variable buffer and the name of a variable record defined in the panel group for the open application.
- "Put Dialog Variable (QUIPUTV) API" on page 67 (QUIPUTV) updates the value of one or more dialog variables by specifying variable buffer of the application program and naming the variable record defined in the panel group for the open application.

"User Interface Manager APIs," on page 1 | APIs by category

## Get Dialog Variable (QUIGETV) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Output | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Get Dialog Variable (QUIGETV) API allows a program to obtain the value of one or more dialog variables by specifying the application program's variable buffer and the name of a variable record defined in the panel group for the open application.

### Authorities and Locks

None.

### Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
OUTPUT; CHAR(*)

The program buffer into which dialog variable values are copied. The variable buffer must be large enough to contain all the variables specified in the definition of the variable record. The dialog variables are copied in the order specified in the variable record, which is specified in the variable record name parameter.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer. The buffer must be large enough to contain all the dialog variables in the definition of the variable record, which is specified in the variable record name parameter.

**Variable record name**
INPUT; CHAR(10)

The name of the variable record definition that determines which dialog variables are copied from the application variable pool to the variable buffer. The variable record must be defined in the panel group for the open application.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Put Dialog Variable (QUIPUTV) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Application handle | Input | Char(8) |
| 2 | Variable buffer | Input | Char(*) |
| 3 | Variable buffer length | Input | Binary(4) |
| 4 | Variable record name | Input | Char(10) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: No

The Put Dialog Variable (QUIPUTV) API updates the value of one or more dialog variables by specifying variable buffer of the application program and naming the variable record defined in the panel group for the open application.

## Authorities and Locks

None.

## Required Parameter Group

**Application handle**
INPUT; CHAR(8)

The application handle assigned by the UIM and returned to the application program by the Open Display Application (QUIOPNDA) API or the Open Print Application (QUIOPNPA) API when the application is opened.

**Variable buffer**
INPUT; CHAR(*)

The program buffer from which dialog variable values are copied. The dialog variables are copied in the order specified in the variable record, which is named in the variable record name parameter.

**Variable buffer length**
INPUT; BINARY(4)

The length of the variable buffer. The buffer must be large enough to contain all the dialog variables in the definition of the variable record, which is specified in the variable record name parameter.

**Variable record name ..**
INPUT; CHAR(10)

The name of the variable record that determines which dialog variables are copied from the variable buffer into the application variable pool. The variable record must be defined in the panel group for the open application.

**Error code**
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF3C90 E | Literal value cannot be changed. |
| CPF6A0B E | Application handle &3 not valid. |
| CPF6A0C E | Application domain error for application &1. |
| CPF6A0F E | Previous error occurred while running application &3. |
| CPF6A24 E | Parameter &1 not passed correctly. |
| CPF6A25 E | Return code length of &1 not valid. |
| CPF6A36 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A37 E | Data not correct for dialog variable &4 in panel group &1 in &2. |
| CPF6A38 E | Variable record &4 not defined in panel group. |
| CPF6A39 E | Variable buffer length too small. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Exit Programs

These are the Exit Programs for this category.

# Exit Program for an Action List Option or Pull-Down Field Choice

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Action list or pull-down information | Input | Char(*) |

QSYSINC Member Name: EUIALEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | List name | Input | Char(10) |
| 5 | List entry handle | Input | Char(4) |
| 6 | Option number | Input | Binary(4) |
| 7 | Function qualifier | Input | Binary(4) |
| 8 | Action results | Input | Binary(4) |

Interface Level 2:

| | | | |
|---|---|---|---|
| 9 | Pull-down field name | Input | Char(10) |

QSYSINC Member Name: EUIALEX

An exit program can be specified for an action list option using the USREXIT attribute of the list action (LISTACT) tag. The exit program is called immediately after the action specified on the ENTER, PROMPT, EXTENTER, or EXTPROMPT attribute of the LISTACT tag is processed.

An exit program for a pull-down field choice can be specified using the USREXIT attribute of the pull-down field choice (PDFLDC) tag. The exit program is called immediately after the action specified on the ACTION attribute of the PDFLDC tag is processed.

Normally, this exit program adds, updates, or removes a list entry for the application when the action list option or pull-down choice action is a command string. The exit program should not change list entries other than the one currently being processed. If it does, remaining UIM list processing may not perform as expected. If the exit program changes the current entry for the list by adding a new entry to the list, the exit program should reset the current entry to the one currently being processed before returning control to the UIM. If the exit program does not do this, remaining UIM list processing may not perform as expected.

Escape messages received by the UIM from the exit program cause the UIM to stop list processing, and display the messages when the panel is reshown. If the dialog variable identifying the exit program is null (for a pointer variable) or blanks (for a character variable), no exception is reported by the UIM and processing continues as if the USREXIT attribute is not specified.

The cancel and exit flags for the job are not reset before the exit program is called, and the flags are not checked after the exit program returns control to the UIM. Therefore, the cancel and exit flags should not be turned on by the exit program.

## Authorities and Locks

None.

## Required Parameter Group

**Action list or pull-down information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure" on page 71.

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*5* The exit program is called for an action list option or pull-down choice when ACTFOR=LIST is specified on the PDFLDC tag.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**List name**
INPUT; CHAR(10)

The name of the list that is currently being processed by the UIM.

**List entry handle**
INPUT; CHAR(4)

The handle of the list entry that is currently being processed by the UIM. The following value can be used:

*EXTE* Processes the extended action entry.

**Option number**
INPUT; BINARY(4)

The option number of the list action or pull-down choice that is currently being processed for the specified list entry.

**Function qualifier**
INPUT; BINARY(4)

Indicates the action to be performed. Here are the possible values:

*0* Performs the action specified on the ENTER or EXTENTER attribute of the list action (LISTACT) tag or the ACTION attribute of the PDFLDC tag.
*-10* Performs the action specified on the PROMPT or EXTPROMPT attribute of the LISTACT tag.

**Action results**
INPUT; BINARY(4)

Indicates whether the list action or pull-down choice is successful. Here are the possible values:

*0* The list action or pull-down choice is successful.
*1* The list action or pull-down choice is unsuccessful.

A list action or pull-down choice is considered unsuccessful if any of the following occurs:

- An escape message is sent by the action program or command.
- The cancel flag for the job is set on by the action program or command.
- The exit flag for the job is set on by the action program or command.

Conversely, the list action is successful if none of the above occur.

**Pull-down field name**
INPUT; CHAR(10)

The name of the pull-down field from which the pull-down choice is selected. If the pull-down field does not have a name, this parameter is set to blanks.

When the action being processed is not a pull-down choice, this parameter is set to blanks.

This parameter is available only when the interface level is set to two or greater.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions."

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | CHAR(10) | List name |
| 44 | 2C | CHAR(4) | List entry handle |
| 48 | 30 | BINARY(4) | Option number |
| 52 | 34 | BINARY(4) | Function qualifier |
| 56 | 38 | BINARY(4) | Action results |
| 60 | 3C | CHAR(10) | Pull-down field name |

## Field Descriptions

**Action results.** Indicates whether the list action or pull-down choice is successful. Here are the possible values:

*0* The list action or pull-down choice is successful.
*1* The list action or pull-down choice is unsuccessful.

A list action or pull-down choice is considered unsuccessful if any of the following occurs:

- An escape message is sent by the action program or command.
- The cancel flag for the job is set on by the action program or command.
- The exit flag for the job is set on by the action program or command.

Conversely, the list action is successful if none of the above occur.

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Function qualifier.** Indicates the action to be performed. Here are the possible values:

*0* Performs the action specified on the ENTER or EXTENTER attribute of the list action (LISTACT) tag or the ACTION attribute of the PDFLDC tag.
*-10* Performs the action specified on the PROMPT or EXTPROMPT attribute of the LISTACT tag.

**List entry handle.** The handle of the list entry that is currently being processed by the UIM. The following value can be used:

*EXTE* Processes the extended action entry.

**List name.** The name of the list that is currently being processed by the UIM.

**Option number.** The option number of the list action or pull-down choice that is currently being processed for the specified list entry.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Pull-down field name.** The name of the pull-down field from which the pull-down choice is selected. If the pull-down field does not have a name, this field is set to blanks. When the action being processed is not a pull-down choice, this field is set to blanks. This field is available only when the structure level field is set to two or greater.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*5* The exit program is called for an action list option or pull-down choice when ACTFOR=LIST is specified on the PDFLDC tag.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Exit Program for Application Formatted Data

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Application formatted data information | Input | Char(*) |

QSYSINC Member Name: EUIAFEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | Panel bidirectional orientation | Input | Char(1) |
| 5 | Device code page | Input | Binary(4) |

Interface Level 2:

No additional parameters are required for interface level 2.

The exit program can update the data formatted by the application every time a panel is displayed, and returns control to the UIM through a normal return. The UIM then finishes formatting the panel and displays it. The area formatted by the application is displayed as defined by the contents of the dialog variable.

An application format exit program can be specified on the USREXIT attribute of the application formatted area (APPFMT) tag. This attribute specifies the name of a dialog variable identifying the program to call.

The UIM does not reevaluate conditions after the text area exit program is called; therefore, changing dialog variables that affect conditions has no effect during this display of the panel.

This exit program is called during the formatting of the panel whenever the panel is displayed. If the dialog variable identifying the program is nulls for a pointer variable or blanks for a character variable, no exception is reported and processing continues as if the USREXIT attribute is not specified.

The exit program can send messages to the UIM for display on the panel. To have the messages shown, the exit program must return control to the UIM by sending a CPF6A05 status or escape message. In this case, any messages sent to the UIM by the incomplete list exit program are shown when the panel is displayed. Otherwise, the exit program returns control to the UIM by doing a normal return.

The exit and cancel flags for the job are not reset before the exit program is called, and the flags are not checked after the exit program returns control to the UIM. Therefore, the exit and cancel flags should not be turned on by the exit program.

## Authorities and Locks

None.

## Required Parameter Group

**Application formatted data information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see “Single Parameter Structure” on page 74.

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*7* The exit program is called to update the application formatted data.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**Panel bidirectional orientation**
INPUT; CHAR(1)

Identifies the attribute orientation of the panel group. Here are the possible values:

*N* BIDI=NONE is specified or the default on the panel group (PNLGRP) tag.
*L* BIDI=LTR is specified on the PNLGRP tag.
*R* BIDI=RTL is specified on the PNLGRP tag.

**Device code page**
INPUT; BINARY(4)

The number of the code page of the requester's display device.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions."

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | CHAR(1) | Panel bidirectional orientation |
| 35 | 23 | BINARY(4) | Device code page |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Device code page.** The number of the code page of the requester's display device.

**Panel bidirectional orientation.** Identifies the attribute orientation of the panel group. Here are the possible values:

*N* BIDI=NONE is specified or the default on the panel group (PNLGRP) tag.
*L* BIDI=LTR is specified on the PNLGRP tag.
*R* BIDI=RTL is specified on the PNLGRP tag.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*7* The exit program is called to update the application formatted data.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## CALL Program for a Function Key

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Function key information | Input | Char(*) |

QSYSINC Member Name: EUIFKCL

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | Function key pressed | Input | Binary(4) |

Interface Level 2:

No additional parameters are required for interface level 2.

Function keys can be assigned to the CALL dialog command on the ACTION attribute of the key item (KEYI) tag. When the function key is pressed, the UIM calls the specified program.

The CALL dialog command is assigned to a function key to process requests that are specific to the application. These requests either do not have a command interface or require more knowledge about the current application than can normally be passed through a command.

The application can perform most functions, but the application developer must be aware of the effects of changing dialog variables, displaying other panels, and so on.

### Authorities and Locks

None.

## Required Parameter Group

**Function key information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure."

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*1* Processes a function key.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**Function key pressed**
INPUT; BINARY(4)

The function key that is pressed:

*1-24* A function key (F1-F24) is processed.
*26* The default action when the Enter key is pressed.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions."

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | BINARY(4) | Function key pressed |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Function key pressed.** The function key that is pressed:

*1-24* A function key (F1-F24) is processed.
*26* The default action when the Enter key is pressed.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*1* Processes a function key.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## CALL Program for a Menu Item

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Menu Item information | Input | Char(*) |

QSYSINC Member Name: EUIMICL

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | Menu option | Input | Binary(4) |

Interface Level 2:

No additional parameters are required for interface level 2.

Menu items can be assigned to the CALL dialog command on the ACTION attribute of the menu item (MENUI) tag. When the menu item is selected, the UIM calls the specified program.

The CALL dialog command is assigned to a menu item to process requests that are specific to the application. These requests either do not have a command interface or require more knowledge about the current application than can normally be passed through a command.

The application can perform most functions, but the application developer must be aware of the effects of changing dialog variables, displaying other panels, and so on.

### Authorities and Locks

None.

## Required Parameter Group

**Menu item information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure."

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

2 Processes a menu item.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**Menu option**
INPUT; BINARY(4)

The option number of the menu item that is selected by the user.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions."

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | BINARY(4) | Menu option |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Menu option.** The option number of the menu item that is selected by the user.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

2 Processes a menu item.

Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## CALL Program for an Action List Option or Pull-Down Field Choice

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Action list or pull-down information | Input | Char(*) |

QSYSINC Member Name: EUIALCL

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | List name | Input | Char(10) |
| 5 | List entry handle | Input | Char(4) |
| 6 | Option number | Input | Binary(4) |
| 7 | Function qualifier | Input | Binary(4) |

Interface Level 2:

| | | | |
|---|---|---|---|
| 8 | Pull-down field name | Input | Char(10) |

QSYSINC Member Name: EUIALCL

An action list option can specify the CALL dialog command on the ENTER, EXTENTER, PROMPT, and EXTPROMPT attributes of the list action (LISTACT) tag. When processing an action list, the UIM calls the specified program, sending the information related as parameters.

A pull-down choice can specify the CALL dialog command on the ACTION attribute of the pull-down field choice (PDFLDC) tag. When processing a selected pull-down choice, the UIM calls the specified program, sending the information related as parameters.

The CALL dialog command on an action list or pull-down choice with ACTFOR=LIST on the PDFLDC tag allows the application to handle a request for a single list entry. When processing the list, the UIM calls the program once for each selected entry in the list. Care must be taken in changing other list entries and the current position because the UIM is accessing the same information to perform the remaining action list processing. For example, changing the current position past some selected list entries causes the UIM to not process the skipped ones.

The CALL dialog command on a pull-down choice with ACTFOR=PANEL on the PDFLDC tag allows the application to handle a request related to the panel as a whole. The UIM calls the program only once before redisplaying the panel.

An escape message received by the UIM while performing list processing stops the processing and displays the messages. The action or selection field still contains the option number or selection character and is marked in error.

## Authorities and Locks

None.

## Required Parameter Group

**Action list or pull-down information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure" on page 81.

**Type of call**
INPUT; BINARY(4)

Specifies whether CALL is due to an Action List Option or a Pull-Down field choice. UIM sets one of the following values:

*3* A program is called for action list option processing or pull-down choice processing when ACTFOR=LIST is specified on the PDFLDC tag.
*9* A pull-down choice. A program is called for pull-down choice processing when ACTFOR=PANEL is specified on the PDFLDC tag.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**List name**
INPUT; CHAR(10)

The name of the list that is currently being processed by the UIM. Blanks indicate that the pull-down choice being processed does not operate against a list entry because ACTFOR=PANEL is specified on the PDFLDC tag.

**List entry handle**
INPUT; CHAR(4)

The list entry handle that is currently being processed by the UIM. Hexadecimal zeros indicate that the pull-down choice being processed does not operate against a list entry. The following value can be used:

*EXTE* Processes the extended action entry.

**Option number**
INPUT; BINARY(4)

The option number of the list action or pull-down choice that is being processed.

**Function qualifier**
INPUT; BINARY(4)

Indicates the action to be performed. Here are the possible values:

*0* Performs the action specified on the ENTER or EXTENTER attribute of the LISTACT tag or the ACTION attribute of the PDFLDC tag.
*-10* Performs the action specified on the PROMPT or EXTPROMPT attribute of the LISTACT tag.

**Pull-down field name**
INPUT; CHAR(10)

The name of the pull-down field from which the pull-down choice is selected. If the pull-down field does not have a name, this parameter is set to blanks. This parameter is available only when the interface level is set to two or greater.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions."

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | CHAR(10) | List name |
| 44 | 2C | CHAR(4) | List entry handle |
| 48 | 30 | BINARY(4) | Option number |
| 52 | 34 | BINARY(4) | Function qualifier |
| 56 | 38 | CHAR(10) | Pull-down field name |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Function qualifier.** Indicates the action to be performed. Here are the possible values:

*0* Performs the action specified on the ENTER or EXTENTER attribute of the LISTACT tag or the ACTION attribute of the PDFLDC tag.
*-10* Performs the action specified on the PROMPT or EXTPROMPT attribute of the LISTACT tag.

**List entry handle.** The list entry handle that is currently being processed by the UIM. Hexadecimal zeros indicate that the pull-down choice being processed does not operate against a list entry. The following value can be used:

*EXTE* Processes the extended action entry.

**List name.** The name of the list that is currently being processed by the UIM. Blanks indicate that the pull-down choice being processed does not operate against a list entry because ACTFOR=PANEL is specified on the PDFLDC tag.

**Option number.** The option number of the list action or pull-down choice that is being processed.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Pull-down field name.** The name of the pull-down field from which the pull-down choice is selected. If the pull-down field does not have a name, this field is set to blanks. This field is available only when the structure level field is set to two or greater.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** Specifies whether CALL is due to an Action List Option or a Pull-Down field choice. UIM sets one of the following values:

*3* A program is called for action list option processing or pull-down choice processing when ACTFOR=LIST is specified on the PDFLDC tag.
*9* A pull-down choice. A program is called for pull-down choice processing when ACTFOR=PANEL is specified on the PDFLDC tag.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Exit Program for Conditioning Panel Items

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Panel item information | Input | Char(*) |

QSYSINC Member Name: EUICTEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Object name | Input | Char(10) |
| 4 | Library name | Input | Char(10) |
| 5 | Object type | Input | Char(10) |
| 6 | Help module name | Input | Char(32) |
| 7 | Panel name | Input | Char(10) |
| 8 | Condition name | Input | Char(10) |

| | | | |
|---|---|---|---|
| 9 | Return code | Output | Char(1) |

Interface Level 2:

No additional parameters are required for interface level 2.

An exit program may be called during condition evaluation for conditions specified on the EXPR attribute of the COND tag and the LINKWHEN and UNLESSn attributes of the LINK tag.

The exit program can be provided for those instances when the determination of user class or existence of an object is not sufficient.

The exit program is called during the evaluation of conditions on a panel. For hypertext link calls, this program will be run every time. When the CHKPGM function is specified on the COND tag, the EVAL attribute on the COND tag will determine when this program is called. A return code set by the exit program will indicate to UIM whether the CHKPGM function should be evaluated to true or false.

The default for the function will be false if the exit program cannot be called. A message will be displayed at the bottom of the panel in the case of an error on the call to the exit program.

If the exit program is being called to process a COND tag on a menu item in a *MENU object, the multiple parameter interface will always be used. The multiple parameter interface will always be used when processing the the LINKWHEN and UNLESSn attributes on a hypertext link also.

## Authorities and Locks

None.

## Required Parameter Group

**Panel item information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see “Single Parameter Structure” on page 84.

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*12* The exit program is called to evaluate the condition for the given panel group, menu, or hypertext link.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM. This value is set only when a *PNLGRP object is being processed. It is set to blanks when the exit program is called for a *MENU object or LINK tag condition processing.

**Object name**
INPUT; CHAR(10)

The name of the panel group or menu object that is being processed. If the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag, this parameter contains the name of the panel group that contains the help identifier named in the LINK tag.

**Library name**
INPUT; CHAR(10)

The name of the library that contains the object that is being processed. If the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag, this parameter contains the name of the library that contains the panel group named in the object name parameter.

**Object type**
INPUT; CHAR(10)

The type of object (*MENU or *PNLGRP) that is being processed.

**Help module name**
INPUT; CHAR(32)

The name of the help module that is being processed when the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag. When conditions on a menu or panel item are being processed, this parameter is blank.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is being processed. This value is blank when the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag.

**Condition name**
INPUT; CHAR(10)

The name of the condition that is currently being evaluated. This is the name as it was entered on the COND NAME attribute of the panel group source. If the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag, this parameter is set to blanks.

**Return code**
OUTPUT; CHAR(1)

Return code from the exit program, indicating the true or false condition. If a value other than one of those listed below is returned, the condition is set to false.

*1* Condition of the CHKPGM function is true.
*0* Condition of the CHKPGM function is false.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 85.

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Object name |
| 34 | 22 | CHAR(10) | Library name |
| 44 | 2C | CHAR(10) | Object type |
| 54 | 36 | CHAR(32) | Help module name |
| 86 | 56 | CHAR(10) | Panel name |
| 96 | 60 | CHAR(10) | Condition name |
| 106 | 6A | CHAR(1) | Return code |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM. This value is set only when a *PNLGRP object is being processed. It is set to blanks when the exit program is called for a *MENU object or LINK tag condition processing.

**Condition name.** The name of the condition that is currently being evaluated. This is the name as it was entered on the COND NAME attribute of the panel group source. If the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag, this field is set to blanks.

**Help module name.** The name of the help module that is being processed when the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag. When conditions on a menu or panel item are being processed, this field is blank.

**Library name.** The name of the library that contains the object that is being processed. If the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag, this field contains the name of the library that contains the panel group named in the object name parameter.

**Object name.** The name of the panel group or menu object that is being processed. If the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag, this field contains the name of the panel group that contains the help identifier named in the LINK tag.

**Object type.** The type of object (*MENU or *PNLGRP) that is being processed.

**Panel name.** The name of the panel that is being processed. This value is blank when the exit program is called to process the condition on a LINKWHEN or UNLESSn attribute on a LINK tag.

**Reserved.** A reserved field.

**Return code.** Return code from the exit program, indicating the true or false condition. If a value other than one of those listed below is returned, the condition is set to false.

*1* Condition of the CHKPGM function is true.
*0* Condition of the CHKPGM function is false.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*12* The exit program is called to evaluate the condition for the given panel group, menu, or hypertext link.

« Exit program introduced: V4R4

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Exit Program for a Cursor-Sensitive Prompt

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Cursor-sensitive prompt information | Input | Char(*) |

QSYSINC Member Name: EUICSEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | Panel element type | Input | Char(1) |
| 5 | Dialog variable name | Input | Char(10) |
| 6 | List name | Input | Char(10) |
| 7 | List entry handle | Input | Char(4) |

Interface Level 2:

No additional parameters are required for interface level 2.

An exit program for handling a cursor prompt request can be specified on the PROMPT attribute of the data item (DATAI), data item extender (DATAIX), and list column (LISTCOL) tags. This attribute specifies the name of a dialog variable identifying the program to call.

This dialog variable cannot be blanks or nulls. The UIM cannot process a prompt request without the exit program.

Normally, the exit program displays a selection list panel. Such a list panel allows the user to select one or more entries from a list of possible choices. The application then sets the selected choices in the appropriate dialog variables, which are shown when the UIM redisplays the panel where the prompt function was requested.

The exit and cancel flags for the job are not reset before the exit program is called, and the flags are not checked after the exit program returns control to the UIM. Therefore, the exit and cancel flags should not be turned on by the exit program.

## Authorities and Locks

None.

## Required Parameter Group

**Cursor-sensitive prompt information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure" on page 87.

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*8* The exit program is called to provide cursor-sensitive prompting.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**Panel element type**
INPUT; CHAR(1)

One of the following values identify the type of panel element prompt:

*1* Prompts for a data item (DATAI tag).
*2* Prompts for a data item extender (DATAIX tag).
*3* Prompts for a list column (LISTCOL tag).

**Dialog variable name**
INPUT; CHAR(10)

The name of the dialog variable where the cursor was positioned when the prompt function was requested.

**List name**
INPUT; CHAR(10)

The name of the list where the cursor was positioned when the prompt function was requested. This parameter is set only when the panel element type parameter indicates that a LISTCOL tag is prompted.

**List entry handle**
INPUT; CHAR(4)

The list entry handle where the cursor was positioned when the prompt function was requested. This parameter is set only when the panel element type parameter indicates that a LISTCOL tag is prompted.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 88.

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | CHAR(1) | Panel element type |
| 35 | 23 | CHAR(1) | Reserved |
| 36 | 24 | CHAR(10) | Dialog variable name |
| 46 | 2E | CHAR(10) | List name |
| 56 | 38 | CHAR(4) | List entry handle |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Dialog variable name.** The name of the dialog variable where the cursor was positioned when the prompt function was requested.

**List name.** The name of the list where the cursor was positioned when the prompt function was requested. This field is set only when the panel element type field indicates that a LISTCOL tag is prompted.

**List entry handle.** The list entry handle where the cursor was positioned when the prompt function was requested. This field is set only when the panel element type field indicates that a LISTCOL tag is prompted.

**Panel element type.** One of the following values identify the type of panel element prompt:

*1* Prompts for a data item (DATAI tag).
*2* Prompts for a data item extender (DATAIX tag).
*3* Prompts for a list column (LISTCOL tag).

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*8* The exit program is called to provide cursor-sensitive prompting.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

---

# Exit Program for General Panel Checking

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | General panel checking information | Input | Char(*) |

QSYSINC Member Name: EUIPGEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | Function key pressed | Input | Binary(4) |
| 5 | Function key qualifier | Input | Binary(4) |
| 6 | Option number | Input | Binary(4) |

Interface Level 2:

| | | | |
|---|---|---|---|
| 7 | Pull-down field name | Input | Char(10) |

QSYSINC Member Name: EUIPGEX

A general exit program may be specified on the USREXIT attribute of the panel definition (PANEL) tag. This attribute specifies the name of a dialog variable identifying the program to call.

The UIM calls the program:

- After any necessary validity checking is done by the UIM

  For a VARUPD=YES attribute on the key item (KEYI) or pull-down field choice (PDFLDC) tags, the UIM has already performed validity checking on all displayed values. If errors are found, the UIM does not call the exit program. The current panel is automatically redisplayed with the appropriate message and error highlighting.

  For a VARUPD=NO attribute, the UIM saves all values from entry fields in their displayed form and does not update the dialog variables. The exit program does not have access to these values.
- After the UIM determines what function to perform

  For some dialog commands, the UIM does not call the exit program. For more information about when the UIM calls the exit program, see Application Display Programming manual.

  Any errors detected during function determination cause the UIM to redisplay the panel without calling the exit program.
- Before the UIM performs the determined function, including dialog variable substitution in command strings specified in the CMD dialog command.

The UIM calls the general exit program and passes information about the function. The exit program performs its task and returns control to the UIM through a normal return or by sending a CPF6A02 or CPF6A03 status message. These messages are transient messages.

If the exit program returns control by sending a CPF6A02 status or escape message, the UIM does not perform the determined function. The panel is redisplayed and any previous messages sent to the UIM are shown.

If the exit program returns control by sending a CPF6A03 status or escape message, the UIM continues performing the determined function. Any previous messages sent to the UIM by the general exit program are shown if and when the function completes. These messages are transient messages.

If the exit program returns control by sending any escape message other than CPF6A02 or CPF6A03, the UIM does not perform the determined function. The panel is redisplayed and any previous messages sent to the UIM are shown. These messages are user messages.

If the exit program returns control normally or sends a non-escape message other than CPF6A02 or CPPF6A03, the UIM continues performing the determined function.

Messages sent to the UIM are displayed only once. They are cleared when the next dialog command is performed, excluding the HELP, PRINT, and PAGEUP or PAGEDOWN (only if paging the message line)

dialog commands. If the exit program marks dialog variables in error, the exit program or other application code is responsible for resetting the dialog variables when the variable is no longer incorrect.

The general exit program allows applications to perform extended validity checking on field values. However, other functions are possible because the exit program has access to panel information used by the UIM to perform a specified function. The exit program can prevent the UIM from processing the function. For example, the exit program could intercept a function key, perform some other function, and tell the UIM to not process the function key. While the UIM does not prevent this type of thing, it is the application developer's responsibility to understand enough about UIM processing to ensure that things work properly.

The action performed by the UIM is determined before the exit program is called. Some actions, such as whether or not to perform the action list processing, depend on the contents of dialog variables or list entries. Changes to these variables or list entries do not cause the UIM to evaluate the determined action again.

Dialog variable substitution into command strings is not done until after the exit program is called; changing dialog variables can affect the final, submitted command.

The exit and cancel flags for the job are not reset before the exit program is called, and they are not checked after the exit program returns control to the UIM. Therefore, exit and cancel flags should not be turned on by the exit program.

## Authorities and Locks

None.

## Required Parameter Group

**General panel checking information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure" on page 91.

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*4* Processes a general panel exit.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**Function key pressed**
INPUT; BINARY(4)

The function key that is pressed:

*1-24* A function key (F1-F24) is pressed.
*26* The Enter key is pressed.
*28* The Page Up key is pressed.

*29* The Page Down key is pressed.
*31* The Home key is pressed.

The general exit program is not called for any other keys.

**Function key qualifier**
INPUT; BINARY(4)

Provides information about the function that will be performed unless the exit program returns through a CPF6A02 message:

*-1* Submits a command from the command line.
*-2* Performs list action processing. The ACTOR attribute of the list area (LIST) tag determines whether the processing is to be performed by the UIM or by the application program displaying the panel.
*-3* Processes a menu item selection.
*-4* The Enter key is pressed and there is no function for the UIM to perform. The default enter action specified on the ENTER attribute of the display panel (PANEL) tag will be performed.
*-5* Processes a cursor-sensitive prompt function.
*-6* Processes a selection from a pull-down choice. This value is used only when the interface level is two or greater.
*-7* Processes the default selection action specified on the SELECT attribute of the PANEL tag because the Enter key was pressed or the user selected one or more entries in an action list or selection list, and there was no function for the UIM to perform.

**Option number**
INPUT; BINARY(4)

When the parameter for the function key qualifier indicates that the UIM is processing a menu item, this parameter is set to the option number of the menu item selected by the user.

When the parameter for the function key qualifier indicates that the UIM is processing a pull-down choice, this parameter is set to the option number of the pull-down choice selected by the user.

For all other values for the function key qualifier, this parameter is not set.

**Pull-down field name**
INPUT; CHAR(10)

The name of the pull-down field from which the pull-down choice is selected. If the pull-down field does not have a name, this parameter is set to blanks.

For all other values of the function key qualifier, this parameter is not set.

This parameter is available only when the interface level is set to two or greater.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 92.

| Offset | | Type | Field |
|---|---|---|---|
| **Dec** | **Hex** | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | BINARY(4) | Function key pressed |

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 38 | 26 | BINARY(4) | Function key qualifier |
| 42 | 2A | BINARY(4) | Option number |
| 46 | 2E | CHAR(10) | Pull-down field name |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Function key pressed.** The function key that is pressed:

*1-24* A function key (F1-F24) is pressed.
*26* The Enter key is pressed.
*28* The Page Up key is pressed.
*29* The Page Down key is pressed.
*31* The Home key is pressed.

The general exit program is not called for any other keys.

**Function key qualifier.** Provides information about the function that will be performed unless the exit program returns through a CPF6A02 message:

*-1* Submits a command from the command line.
*-2* Performs list action processing. The ACTOR attribute of the list area (LIST) tag determines whether the processing is to be performed by the UIM or by the application program displaying the panel.
*-3* Processes a menu item selection.
*-4* The Enter key is pressed and there is no function for the UIM to perform. The default enter action specified on the ENTER attribute of the display panel (PANEL) tag will be performed.
*-5* Processes a cursor-sensitive prompt function.
*-6* Processes a selection from a pull-down choice. This value is used only when the interface level is two or greater.
*-7* Processes the default selection action specified on the SELECT attribute of the PANEL tag because the Enter key was pressed or the user selected one or more entries in an action list or selection list, and there was no function for the UIM to perform.

**Option number.** When the field for the function key qualifier indicates that the UIM is processing a menu item, this field is set to the option number of the menu item selected by the user.

When the field for the function key qualifier indicates that the UIM is processing a pull-down choice, this field is set to the option number of the pull-down choice selected by the user.

For all other values for the function key qualifier, this field is not set.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Pull-down field name.** The name of the pull-down field from which the pull-down choice is selected. If the pull-down field does not have a name, this field is set to blanks.

For all other values of the function key qualifier, this field is not set.

This field is available only when the field for the structure level is set to two or greater.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*4* Processes a general panel exit.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Exit Program for an Incomplete List

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Incomplete list information | Input | Char(*) |

QSYSINC Member Name: EUIILEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | List name | Input | Char(10) |
| 4 | Incomplete list direction | Input | Binary(4) |
| 5 | Number of entries required | Input | Binary(4) |

Interface Level 2:

No additional parameters are required for interface level 2.

An exit program for handling an incomplete list is defined using the Set List Attributes (QUISETLA) API. A dialog variable is specified identifying the program to call.

The dialog variable cannot be blanks or nulls. The UIM cannot process an incomplete list without the exit program.

This exit allows an application to display part of a list without having to build the entire list. It must either add more entries or mark the list as complete in the direction being extended. If the list is not marked complete and fewer entries than the minimum requested by the UIM are added, the exit program is called again. If the list is not marked complete and no entries are added by the exit program, a CPF6A95 error is reported to the program that called the Display Panel (QUIDSPP), Get List Entry (QUIGETLE), Get List Multiple Entries (QUIGETLM), or Remove List Entry (QUIRMVLE) API.

The exit program can send messages to the UIM for display on the panel. To have the messages shown, the exit program must return control to the UIM by sending a CPF6A05 status or escape message. In this case, any messages sent to the UIM by the incomplete list exit program are shown when the panel is displayed. Otherwise, the exit program returns control to the UIM by doing a normal return.

Sometimes the exit program for an incomplete list is called when no panel is displayed; such a call occurs when the QUIGETLE, the QUIGETLM, or the QUIRMVLE API is used. If no panel is displayed when the exit program sends a CPF6A05 status or escape message, any messages sent to the UIM by the exit program are moved to the program message queue for the program calling that API.

The exit and cancel flags for the job are not reset before the exit program is called, and the flags are not checked after the exit program returns control to the UIM. Therefore, the exit and cancel flags should not be turned on by the exit program.

## Authorities and Locks

None.

## Required Parameter Group

**Incomplete list information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see "Single Parameter Structure."

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*6* Processes an incomplete list exit program.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**List name**
INPUT; CHAR(10)

The name of the list that is currently being processed by the UIM.

**Incomplete list direction**
INPUT; BINARY(4)

The direction from the current list entry in which additional entries must be added. Here are the possible values:

*0* More list entries are required after the current list entry.
*1* More list entries are required before the current list entry.

**Number of entries required**
INPUT; BINARY(4)

The minimum number of entries that must be added to the list.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 95.

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Reserved |
| 34 | 22 | CHAR(10) | List name |
| 44 | 2C | BINARY(4) | Incomplete list direction |
| 48 | 30 | BINARY(4) | Number of entries required |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Incomplete list direction.** The direction from the current list entry in which additional entries must be added. Here are the possible values:

*0* More list entries are required after the current list entry.
*1* More list entries are required before the current list entry.

**List name.** The name of the list that is currently being processed by the UIM.

**Number of entries required.** The minimum number of entries that must be added to the list.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Type of call.** This field must be set to the following value:

*6* Processes an incomplete list exit program.

« Exit program introduced: V2R2

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Exit Program for Text Area Data

»

Required Parameter Group for Single Parameter Interface:

| | | | |
|---|---|---|---|
| 1 | Text area data information | Input | Char(*) |

QSYSINC Member Name: EUITAEX

Required Parameter Group for Multiple Parameter Interface:

Interface Level 1:

| | | | |
|---|---|---|---|
| 1 | Type of call | Input | Binary(4) |
| 2 | Application handle | Input | Char(8) |
| 3 | Panel name | Input | Char(10) |
| 4 | Function key pressed | Input | Binary(4) |
| 5 | Function key qualifier | Input | Binary(4) |
| 6 | Text area width | Input | Binary(4) |
| 7 | Text area depth | Input | Binary(4) |
| 8 | Panel bidirectional orientation | Input | Char(1) |
| 9 | Device code page | Input | Binary(4) |

Interface Level 2:

No additional parameters are required for interface level 2.

The exit program can update the data in a text area every time a panel is displayed and returns control to the UIM through a normal return. The UIM will then finish formatting the panel and display it. The area formatted by the application is displayed as defined by the contents of the dialog variable or by the data pointed to by the dialog variable.

A text area exit program can be specified on the USREXIT attribute of the text area (TEXT) tag. This attribute specifies the name of a dialog variable identifying the program to call. The UIM calls the program when:

- The panel is displayed or redisplayed using display panel UIM functions.
- The user presses a key assigned to the PAGEUP, PAGEDOWN, MOVEtop, or ENTER dialog commands and the cursor position is such that the text area should be scrolled.

If the dialog variable identifying the program to call is null for a pointer variable or blanks for a character variable, escape message CPF6A14 is sent to the calling program. If the exit program returns control normally, any messages sent to the UIM by the exit program are displayed on the message line.

The UIM does not reevaluate conditions after the text area exit program is called; therefore, changing dialog variables that affect conditions has no effect during this display of the panel.

The exit and cancel flags for the job are not reset before the exit program is called and the flag are not checked after the exit program returns control to the UIM. Therefore, the exit and cancel flags should not be turned on by the exit program.

## Authorities and Locks

None.

## Required Parameter Group

**Text area data information**
INPUT; CHAR(*)

A structure that provides information related to the user request. For details, see “Single Parameter Structure” on page 97.

**Type of call**
INPUT; BINARY(4)

This parameter must be set to the following value:

*11* The exit program is called to update the text area data.

**Application handle**
INPUT; CHAR(8)

The application handle of the application that is currently being processed by the UIM.

**Panel name**
INPUT; CHAR(10)

The name of the panel that is currently being processed by the UIM.

**Function key pressed**
INPUT; BINARY(4)

The function key that is pressed:

*0* A function key other than F1-F24 is pressed.
*1-24* A function key (F1-F24) is pressed.

**Function key qualifier**
INPUT; BINARY(4)

Provides information about why the user exit program is being called.

*0* The exit program is called for the initial display or redisplay of the panel.
*-1* A key assigned to the PAGEUP dialog command is pressed.
*-2* A key assigned to the PAGEDOWN dialog command is pressed.
*-3* A key assigned to the MOVEtop dialog command is pressed.
*-4* A key assigned to the ENTER dialog command is pressed.

**Text area width**
INPUT; BINARY(4)

The width of the text area within this panel to be formatted.

**Text area depth**
INPUT; BINARY(4)

The depth of the text area within this panel to be formatted.

**Panel bidirectional orientation**
INPUT; CHAR(1)

Identifies the attribute orientation of the panel group. Here are the possible values:

*N* BIDI=NONE is specified or the default on the panel group (PNLGRP) tag.
*L* BIDI=LTR is specified or the default on the PNLGRP tag.
*R* BIDI=RTL is specified or the default on the PNLGRP tag.

**Device code page**
INPUT; BINARY(4)

The number for the code page of the requester's display device.

## Single Parameter Structure

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 98.

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Structure level |
| 4 | 4 | CHAR(8) | Reserved |
| 12 | C | BINARY(4) | Type of call |
| 16 | 10 | CHAR(8) | Application handle |
| 24 | 18 | CHAR(10) | Panel name |
| 34 | 22 | BINARY(4) | Function key pressed |
| 38 | 26 | BINARY(4) | Function key qualifier |
| 42 | 2A | BINARY(4) | Text area width |
| 46 | 2E | BINARY(4) | Text area depth |
| 50 | 32 | CHAR(1) | Panel bidirectional orientation |
| 51 | 33 | BINARY(4) | Device code page |

## Field Descriptions

**Application handle.** The application handle of the application that is currently being processed by the UIM.

**Device code page.** The number for the code page of the requester's display device.

**Function key pressed.** The function key that is pressed:

*0* A function key other than F1-F24 is pressed.
*1-24* A function key (F1-F24) is pressed.

**Function key qualifier.** Provides information about why the user exit program is being called.

*0* The exit program is called for the initial display or redisplay of the panel.
*-1* A key assigned to the PAGEUP dialog command is pressed.
*-2* A key assigned to the PAGEDOWN dialog command is pressed.
*-3* A key assigned to the MOVEtop dialog command is pressed.
*-4* A key assigned to the ENTER dialog command is pressed.

**Panel bidirectional orientation.** Identifies the attribute orientation of the panel group. Here are the possible values:

*N* BIDI=NONE is specified or the default on the panel group (PNLGRP) tag.
*L* BIDI=LTR is specified or the default on the PNLGRP tag.
*R* BIDI=RTL is specified or the default on the PNLGRP tag.

**Panel name.** The name of the panel that is currently being processed by the UIM.

**Reserved.** A reserved field.

**Structure level.** The interface level supported by this structure, indicating which fields and values are available for the current interface level.

**Text area depth.** The depth of the text area within this panel to be formatted.

**Text area width.** The width of the text area within this panel to be formatted.

**Type of call.** This field must be set to the following value:

*11* The exit program is called to update the text area data.

« Exit program introduced: V3R1

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Concepts

These are the concepts for this category.

## Using the User Interface Manager APIs

When calling the user interface manager (UIM) APIs, the calling program must pass arguments by reference. That is, all UIM API programs expect the calling program to pass a space pointer to each of the argument values. For some HLL compilers, this is the only way of passing arguments when calling a program.

Unless otherwise noted, all argument values must be in uppercase, and left-adjusted with trailing blanks. For example, if a program accepts a special value of "ALL" for a parameter defined as CHAR(4), "ALL" must be passed (without the quotes).

## Terms and Definitions

**Application variable pool.** The set of all dialog variable values for an open application.

**Argument list.** In UIM, this list consists of values that are passed to a program.

**Coded character set identifier (CCSID).** A 32-bit number identifying a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other relevant information that uniquely identifies the coded graphic character representation used.

**Contextual help.** Help information about a single item, such as the field on which the cursor is positioned when help is requested.

**Dialog variable.** A named element in a panel group used to pass data values between programs or between a program and a user. The current contents of all dialog variables corresponding to variables in the list are saved as each entry is added.

**Error variable.** The dialog variable specified on the ERRVAR attribute of the variable definition (VAR) tag. The error variable is used to set and test the error status of the dialog variable named on the NAME attribute of the VAR tag.

**Extended action entry.** The first line below the column headings in a list. This line contains entry-capable fields for the option column and for at least one additional list column.

**Extended action list area.** A list area of a panel that contains an extended action entry.

**Extended help.** Help information for all the items on the display; it contains all contextual help items and can contain additional information as well.

**List entry handle.** A value that uniquely distinguishes an entry in a UIM list until it is removed from the list. A list entry handle is meaningful only for a particular open application, list, and entry combination. It has no meaning in any other open UIM application, or even in the same application if the list or the entry is deleted and then re-created. Unpredictable results are possible if a list entry handle is used outside of this definition.

**Message reference key.** A unique string of characters that identify a particular instance of a message in a queue. The message key can also be used to refer to a specific instance of a message in order to move, receive, reply to, resend, or move it.

**Open data path (ODP).** A control block containing information about the merged file attributes and information returned by I/O operations.

**Pop-up window.** An area of the screen with visible borders, which supplements the dialog occurring in the full- screen panel or in a previous pop-up window.

**Pull-down field choice.** A choice that appears in a pull-down menu.

**Trimming.** An operation performed by removing a list entry from the end of the list opposite from the end where the new entry is added.

**Variable buffer.** A buffer used to pass dialog variables between the application program and the UIM.

**Variable record.** A named element of a panel group that identifies the content and layout of a buffer of dialog variables.

**Window.** An area on the display that is treated as a separate display. Windows have visible boundaries and appear to overlay the display from which they are requested.

Top | "User Interface Manager APIs," on page 1 | APIs by category

## Using the User Interface Manager Exit Programs

Through panel definitions and APIs, an application program can cause the user interface manager (UIM) to call other application programs as part of normal panel management. These programs are divided into two classes: programs called through the CALL dialog command and programs called as exit programs.

This section describes the different exits and CALL situations. The UIM does not require a program to follow prescribed actions in its operating environment; attempts to use these exit programs for purposes other than the intended purpose may cause problems for applications and their users.

The CALL dialog command is provided so that an application can link programs and panels without a command interface.

The CALL dialog command causes program calls to handle one of the following requests:

- Function keys
- Menu items
- Action list options
- Pull-down field choices

Exit programs allow applications to perform functions for which the UIM does not provide generic support. Exit program capability exists for the following purposes:

- Action list update processing
- Application formatting of data

- Conditioning panel items
- Cursor-sensitive prompting
- General panel checking
- Incomplete list processing
- Text area data

## Calling a UIM Exit Program

All programs are specified by naming a dialog variable, which identifies the program to call. For the incomplete list exit program, the dialog variable name is specified on the call to the Set List Attributes (QUISETLA) API. For all other exit and CALL programs, the dialog variable name is specified as a parameter to the CALL dialog command in the tag language source.

The UIM uses the current value of the dialog variable to identify the program to call. Depending on the definition of the dialog variable, there are three different ways to call the program:

**Call by address**
When the dialog variable specified is defined with BASETYPE=PTR on the class definition (CLASS) tag, the program is called by address.

It is the responsibility of the application program to ensure that the pointer is set correctly.

**Call by name**
When the dialog variable specified is defined with BASETYPE='CHAR 20' on the CLASS tag, the program is called by name. It is the responsibility of the application program to ensure that the variable contains the object and library name of the program called.

**Extended program model (EPM) call**
When the dialog variable specified is defined with BASETYPE='CHAR 130' on the CLASS tag, the program is called in the extended program model (EPM) environment. This type of call is used when the target program is written in an EPM language such as Pascal or ILE C. It is the responsibility of the application program to ensure that the variable contains the appropriate information for calling an entry point for an EPM environment.

The UIM can call programs in the Integrated Language Environment® (ILE) model using call by address or call by name. Only the main entry point of the program can be used.

## Using a Parameter Interface

When a UIM application is opened, the calling program must specify whether all application programs, called as an exit program or using the CALL dialog command, are passed a single or multiple parameter interface.

When a single parameter is passed, that parameter is a space pointer to a structure containing information for the type of exit or CALL function.

The single parameter interface is used by programs written in languages that can efficiently process data structures. Passing a single parameter is more efficient than passing multiple parameters, and any extensions to the interface are accessible without changing the parameter list of the program being called.

When the single parameter interface is chosen, the structure passed contains a field indicating the level of the structure. Beginning in Operating System/400® (OS/400®) V2R2, the structure level is set to two. All parameters and values of a structure are passed when the structure level is equal to or greater than one, unless the parameter description notes otherwise. For more information about choosing the interface level for exit programs, see "Open Display Application (QUIOPNDA) API" on page 3 and "Open Print Application (QUIOPNPA) API" on page 6.

The interface with multiple parameters is available for programs written in languages that cannot efficiently process data structures. When multiple parameters are passed, each parameter is a space pointer to a piece of information necessary for the type of exit or CALL function.

When the multiple parameter interface is chosen by the program opening the UIM application, the program must also choose which interface level to use. The **interface level** defines the number of parameters passed for a given type of exit or CALL function. This interface allows the parameter interfaces to be extended in the future with upward compatibility for existing application exit programs.

Because the application handle is passed, programs can use dialog variables to access common information. It is more efficient to place one pointer variable that points to a structure or array into the variable pool than to place all the variables from the structure or array into the variable pool.

## Handling Messages

After the UIM calls an application program or runs a control language (CL) command, it handles messages sent to the UIM. There are two kinds of messages: **transient** and **user**.

### Transient

Messages sent by the general exit program are moved to an internal UIM message queue if the exit program returns control to the UIM by sending CPF6A02 or CPF6A03 as a status or escape message. Escape messages are changed to diagnostic messages when they are moved from one message queue to another. These messages are referred to as **transient** messages. When the panel is redisplayed by the UIM and the exit program returns control to the UIM as described above, the transient messages are shown on the message line of the panel. Any escape message will cancel the action as CPF6A02 does, but messages will become user messages not transient. When the message is seen on the panel and the to program is QUIMGFLW, then the message is transient; otherwise, it's a user message.

When certain dialog commands are subsequently performed for the panel, in most cases, transient messages are removed from the UIM's internal message queue and no longer appear in the job log. These messages are not removed if the next dialog command performed is HELP, PRINT, PAGEUP, or PAGEDOWN (if paging the message line).

### User

Messages sent by other types of exit programs or by a CL command are moved to the call message queue identified on the call to the Display Panel (QUIDSPP) API. This is also true if the general exit program returns control to the UIM by sending any escape message, other the CPF6A02 or CPF6A03. Escape messages are changed to diagnostic messages when they are moved from one message queue to another. These messages are referred to as **user** messages. When the panel is redisplayed by the UIM, the user messages are shown on the message line of the panel.

When certain dialog commands are subsequently performed for the panel, the UIM no longer displays the user messages on the panel, see the Application Design Programming  manual. The UIM never changes or removes messages from the call message queue. It is the responsibility of the application program to remove these messages from the call message queue as appropriate.

The following table shows how the UIM handles messages sent by exit programs.

**Messages and exit programs**

| Exit program | Action when messages are sent to the UIM | Escape message difference |
|---|---|---|
| CALL Program for a Function Key | All messages[1] sent by the exit program become **user** messages. | None |

| Exit program | Action when messages are sent to the UIM | Escape message difference |
|---|---|---|
| CALL Program for a Menu Item | All messages[1] sent by the exit program become **user** messages. | None |
| CALL Program for Action List Option and Pull-Down Field Choice | All messages[1] sent by the exit program become **user** messages. | Escape messages stop action or choice processing and the panel is redisplayed. The option or choice is highlighted and any user messages are displayed. |
| Exit Program for General Panel Checking | All messages[1] sent by the exit program become **transient** messages only if the exit program returns control to the UIM by sending CPF6A02 or CPF6A03 as a status or escape message. | CPF6A02 ends the user action. Other messages sent are displayed as transient messages. CPF6A03 does not end the user action. Other messages sent are displayed as transient messages. Escape messages other than CPF6A02 and CPF6A03 ends the user action. Other messages sent are displayed as user messages. See the section for this exit program for details. |
| Exit Program for Action List Option or Pull-Down Field Choice | All messages[1] sent by the exit program become **user** messages. | Escape messages stop action or choice processing and the panel is redisplayed. The option or choice is highlighted and any user messages are displayed. |
| Exit Program for Incomplete List | All messages[1] sent by the exit program become **user** messages only if the exit program returns control to the UIM by sending CPF6A05 as a status or escape message. | If an escape message other than CPF6A05 is sent the UIM cannot complete the list and will send CPF6A95 as an escape message to the application. |
| Exit Program for Application Formatted Data | All[1] but escape messages sent by the exit program are ignored. | Escape messages cause the UIM to not call this exit program again until another display panel is done for the panel. Dialog variables pointing to the exit program have to be set again. The escape message will be displayed on the panel as a user message. |
| Exit Program for a Cursor-Sensitive Prompt | All messages[1] sent by the exit program become **user** messages. | None |
| Exit Program for Text Area Data | All messages[1] sent by the exit program become **user** messages. | CPF6A07 causes the UIM to end the display of this text area panel and return to the display panel caller. See the section for this exit program for details. |
| [1] Messages refer to ESCAPE, INFO, COMP, and DIAG messages. | | |

Escape messages and other accompanying messages are displayed on the panel message line. If the program is sent an escape message to be resent to the UIM, it needs to do the following:

1. Receive the escape message and save its message reference key.
2. Move all INFO, COMP, and DIAG messages to the previous program message queue.
3. Perform any cleanup processing required for the application program.
4. Resend the escape message using its message reference key.

Whenever an escape message is sent to the UIM from the program or a CL command, the message indicates that the function performed was unsuccessful. The UIM then takes whatever action is appropriate for the unsuccessful function. Usually this will be a redisplay of the panel.

Top | "User Interface Manager APIs," on page 1 | APIs by category

# Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Programming interface information

This API descriptions publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced 36
Advanced Function Presentation
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
AS/400
BCOCA
C/400
COBOL/400
Common User Access
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
Enterprise Storage Server
eServer
FlashCopy
GDDM
i5/OS
IBM
IBM (logo)
InfoColor
Infoprint
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
Lotus
Lotus Notes
MO:DCA
MVS
Net.Data
NetServer
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
POWER5+
PowerPC
Print Services Facility
PrintManager
PROFS
RISC System/6000
RPG/400
RS/6000

SAA
SecureWay
SOM
System i
System i5
System Object Model
System/36
System/38
System/390
TotalStorage
VisualAge
WebSphere
xSeries
z/OS

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

## Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER

EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA