System i

Programming
Registration Facility APIs

Version 6 Release 1

System i

Programming
Registration Facility APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in “Notices,” on page 33.

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

Registration Facility APIs

The registration facility APIs provide the capability to:

- Register and deregister exit points with the registration facility.
- Add and remove exit programs to and from the registration facility repository.
- Retrieve exit point and exit program information from the repository.
- Designate the order in which exit programs are called.

Before using the registration facility APIs and registration facility preprocessing exit program, read "Using Registration Facility APIs and Registration Facility" on page 31.

The registration facility APIs are:

- "Add Exit Program (QUSADDEP, QusAddExitProgram) API" (QUSADDEP, QusAddExitProgram) adds an exit program entry to a specific exit point or replaces an existing exit program.
- "Deregister Exit Point (QUSDRGPT, QusDeregisterExitPoint) API" on page 6 (QUSDRGPT, QusDeregisterExitPoint) removes an exit point and all associated exit programs from the registration facility.
- "Register Exit Point (QUSRGPT, QusRegisterExitPoint) API" on page 8 (QUSRGPT, QusRegisterExitPoint) registers an exit point with the registration facility or updates an exit point.
- "Remove Exit Program (QUSRMVEP, QusRemoveExitProgram) API" on page 14 (QUSRMVEP, QusRemoveExitProgram) removes an exit program entry from a specific exit point.
- "Retrieve Exit Information (QUSRTVEI, QusRetrieveExitInformation) API" on page 15 (QUSRTVEI, QusRetrieveExitInformation) retrieves information about one or more exit points and exit programs.

The registration facility preprocessing exit programs are:

- "Preprocessing Exit Program for Add" on page 26 allows for processing to take place before an exit program is added to an exit point.
- "Preprocessing Exit Program for Remove" on page 27 allows for processing to take place before an exit program is removed from an exit point.
- "Preprocessing Exit Program for Retrieve" on page 29 allows for the exit point provider to store the exit program information.

Top | APIs by category

APIs

These are the APIs for this category.

Add Exit Program (QUSADDEP, QusAddExitProgram) API

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Exit point format name	Input	Char(8)
3	Exit program number	Input	Binary(4)
4	Qualified exit program name	Input	Char(20)
5	Exit program data	Input	Char(*)
6	Length of exit program data	Input	Binary(4)
7	Exit program attributes	Input	Char(*)

8	Error code	I/O	Char(*)

Service Program Name: QUSRGFA1
Default Public Authority: *EXCLUDE
Threadsafe: Yes

The Add Exit Program (OPM, QUSADDEP; ILE, QusAddExitProgram) API adds an exit program entry to a specific exit point or replaces an existing exit program. Each exit point can have a single entry, or multiple entries. The exit program number indicates the sequence in which the exit programs should be run. The exit point provider determines the maximum number of exit programs that are allowed for the exit point. The API does not verify that the exit program exists.

If the exit point to which the exit program is being added does not exist, the registration facility creates the exit point and adds the exit program. This exit point will be considered unregistered until it is explicitly registered with the Register Exit Point API. The Add Exit Program, Remove Exit Program, Retrieve Exit Information, and Deregister Exit Point APIs can be performed against an unregistered exit point. This capability allows exit programs to be added to an exit point that will be supported in the future but is not currently registered with the registration facility.

This API provides support similar to the Add Exit Program (ADDEXITPGM) command.

Authorities and Locks

API Public Authority
*EXCLUDE

Exit Registration Lock
*EXCL

Required Parameter Group

Exit point name
INPUT; CHAR(20)

The exit point name to which the exit program is being added.

Exit point format name
INPUT; CHAR(8)

The format name of the exit point to which the exit program is being added.

Exit program number
INPUT; BINARY(4)

The sequence in which the exit programs are to be run when multiple exit point entries for a specific exit point are retrieved. The valid range is 1 through 2 147 483 647 where the processing sequence is from the lowest number to the highest number. Exit program numbers do not need to be consecutive. The following special values are allowed:

-1 The API assigns the next lowest available number for that specific exit point.
-2 The API assigns the highest available number for that specific exit point.

Qualified exit program name
INPUT; CHAR(20)

The exit program name and library that is being added. The first 10 characters contain the exit program name, and the second 10 characters contain the library name in which the exit program resides. The exit program does not need to exist when it is added to the exit point. A specific library name must be specified. The special values *LIBL and *CURLIB are not supported.

Exit program data
INPUT; CHAR(*)

The exit point provider describes what needs to be supplied for this parameter. It is not an error to supply more information than the exit point calls for. Pointer data will not be preserved, and the API does not perform any validation of this parameter.

Length of exit program data
INPUT; BINARY(4)

The length of the exit program data. The valid length is 0 through 2048.

Exit program attributes
INPUT; CHAR(*)

The specified information for the exit program. Refer to "Exit Program Attribute Keys" on page 4 for more information. Any key not specified will be given the default value. The information must be in the following format:

Number of variable length records
BINARY(4)

Total number of all of the variable length records.

Variable length records
The exit program attributes and their values. Refer to "Format for Variable Length Record" for the format of this field.

Error code
I/O; CHAR(*)

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

Format for Variable Length Record

The following table shows the format for the variable length record. For a detailed description of each field, see "Field Descriptions" on page 4.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of variable length record
4	4	BINARY(4)	Exit program attribute key
8	8	BINARY(4)	Length of data
12	C	CHAR(*)	Data

If the length of the data is longer than the key field's data length, the data is truncated at the right. No message is issued.

If the length of the data is shorter than the key field's data length and the key contains binary data, an error message is issued. If the key does not contain binary data, the field is padded with blanks.

It is not an error to specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Data. The value to which a specific exit program attribute is to be set.

Exit program attribute key. The exit program attribute to be set. Refer to "Exit Program Attribute Keys" for more information.

Length of data. The length of the exit program attribute value.

Length of variable length record. The length of the record.

Exit Program Attribute Keys

The following table shows the valid exit program attribute keys for the key field area of the variable length record. For a detailed description of each field, see "Field Descriptions."

Key	Type	Field
1	CHAR(27)	Qualified message file name and message identifier for exit program description
2	CHAR(50)	Exit program text description
3	BINARY(4)	Exit program data CCSID
4	CHAR(1)	Replace
5	CHAR(1)	Threadsafe
6	CHAR(1)	Multithreaded job action

Field Descriptions

Exit program data CCSID. The coded character set identifier (CCSID) used for working with the exit program data. The default value is 0.

0 Use the current job default CCSID.
CCSID A valid CCSID number. The valid CCSID range is 1 through 65 535 but not 65 534. The CCSID will be validated by the API.

Exit program text description. The text for the exit program description. When this key is specified, the qualified message file name and message identifier for exit program description field must not be specified. The default value is blanks.

Multithreaded job action. The action to take in a multithreaded job. This key has no direct relationship with the threadsafe key; however, the value for the threadsafe key can be used to determine the multithreaded job action. The default value is 0. Valid values for this key are:

0 Use the QMLTTHDACN system value to determine the action to take.
1 Run the exit program in a multithreaded job.
2 Run the exit program in a multithreaded job, but send an informational message. CPI3C80 can be used as the informational message.
3 Do not run the exit program in a multithreaded job. Depending on the exit point, do one of the following:
1. Send an escape message and do not call the exit program. CPF3C80 can be used as the escape message.
2. Send an informational message and do not call the exit program. CPF3C80 can be used as the informational message.
3. Call the exit program in a non-multithreaded job.

If you use the threadsafe value to determine the value for the multithreaded job action, consider the following recommendations:

1. If the threadsafe value is 0, the multithreaded job action should be set to 3.
2. If the threadsafe value is 1, the multithreaded job action should be set to 0.
3. If the threadsafe value is 2, the multithreaded job action should be set to 1.

Qualified message file name and message identifier for exit program description. A message file and message identifier that contains the exit program description. When this key is specified, the exit program text description key must not be specified. The message file and message identifier do not have to exist at the time the exit program is added. The default value is blanks. Refer to "Qualified Message File Format" for more information.

Replace. Whether to replace an existing exit program entry. The combination of the exit program name and exit program number define an exit program entry. The default value is 0. Valid values for this key are:

0 Do not replace an existing exit program entry.
1 Replace an existing exit program entry.

Threadsafe. Whether the exit program entry is threadsafe. This key has no direct relationship with the multithreaded job action key. It is intended for documentation purposes only. The default value is 1. Valid values for this key are:

0 The exit program entry is not threadsafe.
1 The threadsafe status of the exit program entry is not known.
2 The exit program entry is threadsafe.

Qualified Message File Format

The following table shows the layout of the qualified message file name and message identifier for exit program description field. For a detailed description of each field, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Message file name
10	10	CHAR(10)	Message file library name
20	14	CHAR(7)	Message identifie

Field Descriptions

Message file library name. The library name in which the message file resides. The special value *CURLIB is not supported. The possible values are:

**LIBL* Search the library list for the message file. This value uses the first message file in the library list that contains the message identifier.
library name The name of the message library the message file resides in.

Message file name. The name of the message file that contains the exit program text description.

Message identifier. The message identifier for the description.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C81 E	Value for key &1 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C85 E	Value for key &1 not allowed with value for key &2.
CPF3C90 E	Literal value cannot be changed.
CPF3CD2 E	Exit point name &1 not valid.
CPF3CD3 E	Exit point format name &1 not valid.
CPF3CD4 E	Maximum number of exit programs reached for exit point &1 with format &2.
CPF3CD6 E	Length of exit program data &1 not valid.
CPF3CD9 E	Requested function cannot be performed at this time.
CPF3CDA E	Registration facility repository not available for use.
CPF3CDE E	Exit program name &1 library &2 not valid.
CPF3CDF E	Exit program number &1 already assigned for exit point &2 with format &3.
CPF3CE1 E	Exit program number &1 not valid.
CPF3CE5 E	Exit point &1 with format &2 will not allow exit program &3 library &4 to be added.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9802 E	Not authorized to object &2 in &3.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

Top | "Registration Facility APIs," on page 1 | APIs by category

---

Deregister Exit Point (QUSDRGPT, QusDeregisterExitPoint) API

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Exit point format name	Input	Char(8
3	Error code	I/O	Char(*)

Service Program Name: QUSRGFA1
Default Public Authority: *EXCLUDE
Threadsafe: Yes

The Deregister Exit Point (OPM, QUSDRGPT; ILE, QusDeregisterExitPoint) API removes an exit point and all associated exit programs from the registration facility. However, to deregister the exit point, the allow deregistration exit point control must be set to indicate that the exit point is eligible for deregistration.

Authorities and Locks

API Public Authority
*EXCLUDE

Exit Registration Lock
*EXCL

Required Parameter Group

Exit point name
INPUT; CHAR(20)

The exit point name for the exit point being removed. The following can be specified for the exit point name:

*generic**	All exit point names that have names beginning with the generic string.
exit point name	Specific exit point name.

Exit point format name
INPUT; CHAR(8)

The format name for the exit point being removed. The following can be specified for the exit point format name:

*generic**	All exit point format names that have names beginning with the generic string.
exit point format name	Specific exit point format 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
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CD2 E	Exit point name &1 not valid.
CPF3CD3 E	Exit point format name &1 not valid.
CPF3CD9 E	Requested function cannot be performed at this time.
CPF3CDA E	Registration facility repository not available for use.
CPF3CDB E	Exit point &1 with format &2 does not exist.
CPF3CDC E	&1 exit points deregistered. &2 exit points not deregistered.
CPD3CD1 E	Exit point &1 with format &2 not deregistered.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9802 E	Not authorized to object &2 in &3.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

"Deregister Exit Point (QUSDRGPT, QusDeregisterExitPoint) API" on page 6 | "Registration Facility APIs," on page 1 | APIs by category

Register Exit Point (QUSRGPT, QusRegisterExitPoint) API

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Exit point format name	Input	Char(8)
3	Exit point controls	Input	Char(*)
4	Error code	I/O	Char(*)

Service Program Name: QUSRGFA1
Default Public Authority: *EXCLUDE
Threadsafe: Yes

The Register Exit Point (OPM, QUSRGPT; ILE, QusRegisterExitPoint) API registers an exit point with the registration facility. Each exit point can have a single exit program or multiple exit programs associated with it. Each exit point can be registered multiple times with a unique format name. The format name is defined by the exit point provider. The format name can be used to define the structural layout of the exit program data, the number and type of parameters to be passed, and so on. The exit point controls provide information to help manage and control the use of the exit point. The user profile calling the Register Exit Point API does not need to be authorized to the preprocessing exit programs.

Updating of an exit point is performed by reregistering the exit point with new values for the exit point control keys. The registration facility will update the control keys and maintain the current list of exit programs that are associated with the exit point. The following conditions apply to updating the exit point control keys:

- Allow deregister: This control key is set the first time the exit point is registered and cannot be changed.
- Allow change of exit point controls: When this control key is set to 0 (cannot be changed), none of the control keys are eligible to be updated.
- Maximum number of exit programs: Updating this control key to a value less than the number of exit programs currently under the exit point results in an error. The update is not performed.
- Preprocessing exit program information for add function: If the new preprocessing exit program value is not *NONE and the Preprocessing Exit Program for Retrieve is *NONE, the Preprocessing Exit Program for Add is called for each exit program associated with the exit point. If the preprocessing exit program returns to the API the return code to not add an exit program, an error occurs. No update is performed.

  If updating the preprocessing exit program to *NONE and the preprocessing exit program information for retrieve function field is also *NONE, the API updates the control key. If the preprocessing exit program information for retrieve function field is not *NONE, an error is returned and no update is performed.
- Preprocessing exit program information for remove function: If updating the preprocessing exit program to *NONE and the preprocessing exit program information for retrieve function field is also *NONE, the API updates the control key. If the preprocessing exit program for retrieve is not set to *NONE, an error is returned and no update is performed.
- Preprocessing exit program information for retrieve function: When the new value for the preprocessing exit program is not *NONE, preprocessing exit programs for add and remove must be either currently specified for the exit point or must be specified on the registration call. The registration facility calls the Preprocessing Exit Program for Add for each of the exit programs associated with the exit point. The facility then removes these exit programs (without calling the Preprocessing Exit Program for Remove) from the registration facility repository and updates the exit point. If the preprocessing exit program returns to the API the return code to not add an exit program, an error occurs and no update is performed.

  When the new value for the preprocessing exit program is *NONE, the API will change the value. Exit point providers are responsible for moving the exit program information that they stored to the registration facility by using the Add Exit Program API.

- Qualified message file and message identifier for exit point description: The registration facility updates this control key with the new value. When this control key is specified for an update, the text for exit point description control key must not be specified.
- Exit point text description: The registration facility updates this control key with the new value. When this control key is specified for an update, the qualified message file and message identifier for exit point description control key must not be specified.

Unregistered Exit Points

The registration facility creates an exit point when an exit program is requested to be added to an exit point that does not exist. The facility uses the default values for the exit point control keys. This exit point is considered unregistered until it is explicitly registered with this API.

An unregistered exit point that was created by the Add Exit Program API can be registered using the Register Exit Point API. Unregistered exit points and related information can be displayed using the Work with Registration Information (WRKREGINF) command or retrieved using the Retrieve Exit Information API.

The Add Exit Program, Remove Exit Program, Retrieve Exit Information, and Deregister Exit Point APIs can be run against an unregistered exit point. The ability to deregister an unregistered exit point enables the removal of exit points created by the Add Exit Program API in error. For example, if the exit point name specified on the call to the Add Exit Program API were misspelled, the exit point can be deregistered.

When registering an unregistered exit point, the exit point control keys are reset to what is specified on the call to the Register Exit Point API. The following conditions prevent the registration of an unregistered exit point:

- A preprocessing exit program is specified for add. The registration facility calls the Preprocessing Exit Program for Add for each exit program that was added to the unregistered exit point. If an exit program currently listed under the unregistered exit point cannot be added, the preprocessing exit program then notifies the registration facility. When this occurs, the exit point provider must remove the exit program from the unregistered exit point (using the Remove Exit Program API) and must register the exit point again.
- The current number of exit programs associated with the unregistered exit point exceeds the maximum number of exit programs specified when the exit point is registered. When this occurs, the exit point provider should do either of the following:

  – Remove the appropriate number of exit programs from the unregistered exit point (using the Remove Exit Program API)
  – Change the maximum number of exit programs field to a higher value

Authorities and Locks

API Public Authority
*EXCLUDE

Exit Registration Lock
*EXCL

Required Parameter Group

Exit point name
INPUT; CHAR(20)

The exit point name to register. IBM® i5/OS® exit points are named QIBM_Q*ccc_name*, where *ccc* is the component identifier. All other IBM exit points are named QIBM_*wccc_name*, where *w* is a

character A through I and *ccc* is the component identifier. User-supplied exit point names should not preface their exit point names with QIBM. User-supplied exit point names should start with the company name to eliminate most problems involving name uniqueness. An exit point name must be a valid *NAME (basic name) and all uppercase. See ELEM (Element) Statement in the Control language topic collection for more information about *NAME.

Exit point format name
INPUT; CHAR(8)

The format defined by the exit point provider. The format specifies the layout of the exit program data or the parameters to be passed, or both. The exit point format name must be a valid *NAME (basic name) and all uppercase characters.

Exit point controls
INPUT; CHAR(*)

The exit point control fields for managing the exit point. Any field not specified will be given the default value. Refer to "Exit Point Control Keys" on page 11 for more information. The information must be in the following format:

Number of variable length records
BINARY(4)

The total number of all of the variable length records.

Variable length records
The fields of the exit point controls to set. Refer to "Format for Variable Length Record" for more information.

Error code
I/O; CHAR(*)

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

Format for Variable Length Record

The following table shows the layout of the variable length record. For a detailed description of each field, see "Field Descriptions" on page 11.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of variable length record
4	4	BINARY(4)	Exit point control key
8	8	BINARY(4)	Length of data
12	C	CHAR(*)	Data

If the length of the data is longer than the key field's data length, the data is truncated at the right. No message is issued.

If the length of the data is shorter than the key field's data length and the key contains binary data, an error message is issued. If the key does not contain binary data, the field is padded with blanks.

It is not an error to specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Data. The value to which a specific exit point control is to be set.

Exit point control key. The exit point control to be set. Refer to "Exit Point Control Keys" for more information.

Length of data. The length of the exit point control value.

Length of variable length record. The length of the record including this field.

Exit Point Control Keys

The following table shows the valid exit point control keys for the key field area of the variable length record. For a detailed description of each field, see "Field Descriptions."

Key	Type	Field
1	CHAR(1)	Allow deregistration
2	CHAR(1)	Allow change of exit point controls
3	BINARY(4)	Maximum number of exit programs
4	CHAR(28)	Preprocessing exit program information for add function
5	CHAR(28)	Preprocessing exit program information for remove function
6	CHAR(28)	Preprocessing exit program information for retrieve function
7	CHAR(27)	Qualified message file name and message identifier for exit point description
8	CHAR(50)	Exit point text description

Field Descriptions

Allow change of exit point controls. Whether the exit point controls can be changed. When 0 (no change) is specified, the only means of changing the exit point controls is to:

- Deregister the exit point (if allow deregister is set to 1)
- Reregister the exit point
- Add the exit programs again

The default value is 1.

0 The exit point controls cannot be changed.
1 The exit point controls can be changed.

Allow deregistration. Whether the exit point can be deregistered (removed from the registration facility repository). When 0 is specified, the exit point can never be removed from the registration facility repository. This control is set when the exit point is registered and cannot be changed. The default value is 1.

0 The exit point cannot be deregistered.
1 The exit point can be deregistered.

Exit point text description. The text for the exit point description. When this key is specified, the qualified message file name and message identifier for exit point description key must not be specified. The default value is blanks.

Maximum number of exit programs. The number of exit programs that this exit point can have. The minimum number of exit programs is 1. The default value is -1.

-1 No maximum.
>0 The maximum number of exit programs.

Preprocessing exit program information for add function. The format and the exit program that the registration facility calls when the Add Exit Program API is called for the exit point. This program performs any function that is needed by the exit point when an exit program is added to it. The exit program must exist when the exit point is registered. Refer to "Preprocessing Exit Program Format" on page 13 for the format of this field.

Preprocessing exit program information for remove function. The format and the exit program that the registration facility calls when the Remove Exit Program API is called for the exit point. This program performs any function that is needed by the exit point when an exit program is removed from it. The exit program must exist when the exit point is registered. Refer to "Preprocessing Exit Program Format" on page 13 for the format of this field.

Preprocessing exit program information for retrieve function. The format and the exit program that the registration facility calls when the Retrieve Exit Information API is called for the exit point. This exit program cannot be specified without specifying preprocessing exit programs for add and remove. When this exit program is specified, the exit point provider will store all the exit program information instead of the registration facility. The exit program must exist when the exit point is registered. Refer to "Preprocessing Exit Program Format" on page 13 for the format of this field.

Qualified message file name and message identifier for exit point description. A message file and message identifier that contains the exit point description. When this key is specified, the exit point text description control key must not be specified. The message file and message identifier do not have to exist at the time of registration. The default value is blanks. Refer to "Qualified Message File Format" for the format of this field.

Qualified Message File Format

The following table shows the layout of the qualified message file name and message identifier for exit point description field. For a detailed description of each field, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Message file name
10	A	CHAR(10)	Message file library name
20	14	CHAR(7)	Message identifier

Field Descriptions

Message file library name. The library name in which the message file resides. The special value *CURLIB is not supported. The possible values are:

**LIBL* Search the library list for the message file. This value uses the first message file in the library list that contains the message identifier.
library name The name of the message library the message file resides in.

Message file name. The name of the message file that contains the exit point description.

Message identifier. The message identifier for the description.

Preprocessing Exit Program Format

The following table shows the layout of the preprocessing exit program information fields. For a detailed description of each field, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Preprocessing exit program name
10	A	CHAR(10)	Preprocessing exit program library name
20	14	CHAR(8)	Preprocessing exit program format name

Field Descriptions

Preprocessing exit program format name. The format name for the preprocessing exit program. If *NONE is specified for the preprocessing exit program name, this field must be blank. The possible values for the format names follow:

ADDP0100 The required parameter group for the Preprocessing Exit Program for Add.
RMVP0100 The required parameter group for the Preprocessing Exit Program for Remove.
RTVI0100 The required parameter group for the Preprocessing Exit Program for Retrieve.

Refer to "Registration Facility APIs," on page 1 for information about the required parameter group of each preprocessing exit program.

Preprocessing exit program library name. The library name in which the preprocessing exit program resides. If *NONE is specified for the preprocessing exit program name, this field must be blank. The special values *LIBL and *CURLIB are not supported.

Preprocessing exit program name. The name of the preprocessing exit program that is called by the registration facility when the corresponding function is requested for the exit point. The default value is *NONE. The possible values are:

**NONE* No exit program is supplied.
exit program name The exit program name.

If *NONE is specified for the preprocessing exit program name, the library name and format name must be blank.

Error Messages

Message ID	**Error Message Text**
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C81 E	Value for key &1 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C84 E	Key &1 required with value specified for key &2.
CPF3C85 E	Value for key &1 not allowed with value for key &2.
CPF3C90 E	Literal value cannot be changed.

Message ID	Error Message Text
CPF3CD1 E	Exit point &1 with format &2 already registered.
CPF3CD2 E	Exit point name &1 not valid.
CPF3CD3 E	Exit point format name &1 not valid.
CPF3CD4 E	Maximum number of exit programs reached for exit point &1 with format &2.
CPF3CD5 E	Exit point control &1 cannot be changed.
CPF3CD7 E	Preprocessing exit program &1 library &2 with format &3 not valid.
CPF3CD8 E	Registration of exit point &1 with format &2 not performed.
CPF3CD9 E	Requested function cannot be performed at this time.
CPF3CDA E	Registration facility repository not available for use.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9802 E	Not authorized to object &2 in &3.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3

API introduced: V3R1

"Register Exit Point (QUSRGPT, QusRegisterExitPoint) API" on page 8 | "Registration Facility APIs," on page 1 | APIs by category

---

Remove Exit Program (QUSRMVEP, QusRemoveExitProgram) API

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Exit point format name	Input	Char(8)
3	Exit program number	Input	Binary(4)
4	Error code	I/O	Char(*)

Service Program Name: QUSRGFA1
Default Public Authority: *EXCLUDE
Threadsafe: Yes

The Remove Exit Program (OPM, QUSRMVEP; ILE, QusRemoveExitProgram) API removes an exit program entry from a specific exit point that is registered or unregistered. An *unregistered exit point* is an exit point that the registration facility creates at the time an exit program is added if the exit point does not exist.

This API provides support similar to the Remove Exit Program (RMVEXITPGM) command.

Authorities and Locks

API Public Authority
*EXCLUDE

Exit Registration Lock
*EXCL

Required Parameter Group

Exit point name
INPUT; CHAR(20)

The exit point name from which the exit program is being removed.

Exit point format name
INPUT; CHAR(8)

The exit point format name from which the exit program is being removed.

Exit program number
INPUT; BINARY(4)

The exit program number of the exit program to be removed. The following values are allowed:

-1	All exit programs are removed for the exit point and format name specified.
exit program number	The specific exit program number to remove.

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
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CD9 E	Requested function cannot be performed at this time.
CPF3CDA E	Registration facility repository not available for use.
CPF3CDB E	Exit point &1 with format &2 does not exist.
CPF3CDD E	Exit program number &1 does not exist.
CPF3CE1 E	Exit program number &1 not valid.
CPF3CEA E	Exit point &1 with format &2 will not allow exit program &3 library &4 to be removed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9802 E	Not authorized to object &2 in &3.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPI3C03 I	&1 exit programs removed. &2 exit programs not removed.

API introduced: V3R1

"Remove Exit Program (QUSRMVEP, QusRemoveExitProgram) API" on page 14 | "Registration Facility APIs," on page 1 | APIs by category

Retrieve Exit Information (QUSRTVEI, QusRetrieveExitInformation) API

Required Parameter Group:

1	Continuation handle	Input	Char(16)
2	Receiver variable	Output	Char(*)
3	Length of receiver variable	Input	Binary(4)
4	Format name	Input	Char(8)

5	Exit point name	Input	Char(20)
6	Exit point format name	Input	Char(8)
7	Exit program number	Input	Binary(4)
8	Exit program selection criteria	Input	Char(*)
9	Error code	I/O	Char(*)

Service Program Name: QUSRGFA2
Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Exit Information (OPM, QUSRTVEI; ILE, QusRetrieveExitInformation) API retrieves information about one or more exit points and their associated exit programs. This API returns information similar to the Work with Registration Information (WRKREGINF) command.

Authorities and Locks

API Public Authority
*USE

Exit Registration Lock
*SHRNUP

Required Parameter Group

Continuation handle
INPUT; CHAR(16)

The value returned to the user in the receiver variable when only partial exit information is returned. This parameter must be set to blanks on the first call to this API. This parameter is used when more information is available to return than what could fit in the receiver variable. When you specify a continuation handle for this parameter, all other parameters must have the same values as the call to the API that generated the continuation handle. Failure to do so may result in incomplete or inaccurate information.

Entries are only returned in their entirety; the API never returns anything less. If there is not enough space for the entire entry, the continuation handle is set to something other than blanks.

Receiver variable
OUTPUT; CHAR(*)

The variable that is to receive the exit information requested.

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 not be predictable. The minimum length is 8 bytes.

Format name
INPUT; CHAR(8)

The format of the exit information to be returned. You must use one of the following format names:

EXTI0100	Exit point information
EXTI0200	Basic exit program information
EXTI0300	Complete exit program information

Refer to “EXTI0100 Format” on page 18, “EXTI0200 Format” on page 19, and “EXTI0300 Format” on page 20 for more information.

Exit point name
INPUT; CHAR(20)

The name of the exit point for which information is being retrieved. You must use one of the following values.

Note: The specified values in these value descriptions pertain to the exit point format name, exit program number, and exit program selection criteria fields.

**ALL*	All registered and unregistered exit point names that meet the specified values will be returned.
**REGISTERED*	All registered exit point names that meet the specified values will be returned.
**UNREGISTERED*	All unregistered exit point names that meet the specified values will be returned.
*generic**	All registered and unregistered exit point names that have names beginning with the generic string and meet the specified values will be returned.
exit point name	The registered or unregistered exit point name that was specified that meets the specified values will be returned.

Exit point format name
INPUT; CHAR(8)

The exit point format name associated with an exit point. You must use one of the following values.

Note: The specified values in these value descriptions pertain to the exit point name, exit program number, and exit program selection criteria fields.

**ALL*	All exit point format names that meet the specified values will be returned.
*generic**	All exit point format names that have names beginning with the generic string and meet the specified values will be returned.
exit point format name	The exit point format name that was specified that meets the specified values will be returned.

Exit program number
INPUT; BINARY(4)

The number of the exit program. If you specify format EXTI0100, this parameter is ignored. You must use one of the following values.

Note: The specified values in these value descriptions pertain to the exit point name, exit point format name, and exit program selection criteria fields.

-1	All exit programs that meet the specified values will be returned.
exit program number	The exit program number to be returned. The entry must meet the specified values to be returned. The valid range is 1 through 2 147 483 647.

Exit program selection criteria
INPUT; CHAR(*)

The selection criteria to be used when selecting which exit programs associated with the exit point are returned. The comparison data is compared against the exit program data. The comparison data and the exit program data to compare it to must be from 1 through 256 characters, and no CCSID normalization is performed. Using characters from the invariant character set for the comparison data is recommended.

For format EXTI0100, this parameter is ignored.

The information must be in the following format:

Number of selection criteria
BINARY(4)

The total number of selection criteria. Specify 0 if no selection criteria are specified. The maximum value for this field is 1.

Selection criteria array
CHAR(*)

The selection criteria. Refer to "Format for Exit Program Selection Criteria" on page 24 for more information.

Error code
I/O; CHAR(*)

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

EXTI0100 Format

The following information is returned for the EXTI0100 format. This format provides information on an exit point. For a detailed description of each field, see "Field Descriptions" on page 21.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(16)	Continuation handle
24	18	BINARY(4)	Offset to first exit point entry
28	1C	BINARY(4)	Number of exit point entries returned
32	20	BINARY(4)	Length of exit point entry
36	24	CHAR(*)	Reserved
Note: Exit point entry information. These fields are repeated for each exit point entry returned.			
		CHAR(20)	Exit point name
		CHAR(8)	Exit point format name
		BINARY(4)	Maximum number of exit programs
		BINARY(4)	Current number of exit programs
		CHAR(1)	Allow deregistration
		CHAR(1)	Allow change of exit point controls
		CHAR(1)	Registered exit point
		CHAR(10)	Preprocessing exit program name for adding an exit program
		CHAR(10)	Preprocessing exit program library name for adding an exit program
		CHAR(8)	Preprocessing exit program format name for adding an exit program
		CHAR(10)	Preprocessing exit program name for removing an exit program
		CHAR(10)	Preprocessing exit program library name for removing an exit program
		CHAR(8)	Preprocessing exit program format name for removing an exit program
		CHAR(10)	Preprocessing exit program name for retrieving exit information

Offset		Type	Field
Dec	Hex		
		CHAR(10)	Preprocessing exit program library name for retrieving exit information
		CHAR(8)	Preprocessing exit program format name for retrieving exit information
		CHAR(1)	Exit point description indicator
		CHAR(10)	Exit point description message file name
		CHAR(10)	Exit point description message file library name
		CHAR(7)	Exit point description message ID
		CHAR(50)	Exit point text description
		CHAR(*)	Reserved

EXTI0200 Format

The following information is returned for the EXTI0200 format. This format provides basic information on an exit program. The exit programs will be in ascending sequence based on the exit point name, exit point format name, and exit program number. For a detailed description of each field, see "Field Descriptions" on page 21.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(16)	Continuation handle
24	18	BINARY(4)	Offset to first exit program entry
28	1C	BINARY(4)	Number of exit program entries returned
32	20	BINARY(4)	Length of exit program entry
36	24	CHAR(*)	Reserved
Note: Exit program entry information. These fields are repeated for each exit program entry returned.			
		BINARY(4)	Offset to next exit program entry
		CHAR(20)	Exit point name
		CHAR(8)	Exit point format name
		CHAR(1)	Registered exit point
		CHAR(1)	Complete entry
		CHAR(2)	Reserved
		BINARY(4)	Exit program number
		CHAR(10)	Exit program name
		CHAR(10)	Exit program library name
		BINARY(4)	Exit program data CCSID
		BINARY(4)	Offset to exit program data
		BINARY(4)	Length of exit program data
		CHAR(1)	Threadsafe
		CHAR(1)	Multithreaded job action

Offset		Type	Field
Dec	Hex		
		CHAR(1)	QMLTTHDACN system value
		CHAR(1)	Reserved
		CHAR(*)	Reserved
Note: Exit program data			
		CHAR(*)	Exit program data

EXTI0300 Format

The following information is returned for the EXTI0300 format. This format provides complete information on an exit program. The exit programs will be in ascending sequence based on the exit point name, exit point format name, and exit program number. For a detailed description of each field, see "Field Descriptions" on page 21.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(16)	Continuation handle
24	18	BINARY(4)	Offset to first exit program entry
28	1C	BINARY(4)	Number of exit program entries returned
32	20	BINARY(4)	Length of exit program entry
36	24	CHAR(*)	Reserved
Note: Exit program entry information. These fields are repeated for each exit program entry returned.			
		BINARY(4)	Offset to next exit program entry
		CHAR(20)	Exit point name
		CHAR(8)	Exit point format name
		CHAR(1)	Registered exit point
		CHAR(1)	Complete entry
		CHAR(2)	Reserved
		BINARY(4)	Exit program number
		CHAR(10)	Exit program name
		CHAR(10)	Exit program library name
		CHAR(1)	Exit program description indicator
		CHAR(10)	Exit program description message file name
		CHAR(10)	Exit program description message file library name
		CHAR(7)	Exit program description message ID
		CHAR(50)	Exit program text description
		CHAR(2)	Reserved
		BINARY(4)	Exit program data CCSID
		BINARY(4)	Offset to exit program data
		BINARY(4)	Length of exit program data

Offset		Type	Field
Dec	Hex		
		CHAR(1)	Threadsafe
		CHAR(1)	Multithreaded job action
		CHAR(1)	QMLTTHDACN system value
		CHAR(1)	Reserved
		CHAR(*)	Reserved
Note: Exit program data			
		CHAR(*)	Exit program data

Field Descriptions

Allow change of exit point controls. Whether the exit point controls can be changed. The possible values follow:

0 The exit point controls cannot be changed.
1 The exit point controls can be changed.

Allow deregistration. Whether the exit point can be deregistered. The possible values follow:

0 The exit point cannot be deregistered.
1 The exit point can be deregistered.

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

If the continuation handle is set to a value other than blanks, this field contains an approximation of the total bytes available. At a minimum, this field contains the actual number of bytes available.

Bytes returned. The number of bytes of data returned.

Complete entry. Whether the information returned for the exit point is complete and accurate. Incomplete information may occur when an exit point's provider is storing the exit program information instead of having the registration facility store it. The exit point notifies the API that the information it returned to the API is incomplete or inaccurate.

All information for the exit program entry up to this field is complete and accurate. All information for the exit program entry following this field should be ignored.

The possible values follow:

0 The exit point entry information is not complete or accurate.
1 The exit point entry information is complete and accurate.

Continuation handle. The handle that is returned when more data is available to return, but the receiver variable is not large enough. The handle indicates the point in the repository that the retrieval stopped. If the handle is used on the next call to the API, the API returns more data starting at the point that the handle indicates. This field is set to blanks when all information is returned.

Current number of exit programs. The current number of exit programs associated with the exit point.

Exit point description indicator. Whether the exit point description is contained in a message file or text. The possible values follow:

0 The exit point description is contained in a message file.
1 The exit point description is text.

Exit point description message file name. The name of the message file that contains the exit point description. This field will contain blanks when a text description is provided for the exit point description.

Exit point description message file library name. The name of the library in which the exit point description message file resides. This field will contain blanks when a text description is provided for the exit point description.

Exit point description message ID. The message identifier for the exit point description. This field will contain blanks when a text description is provided for the exit point description.

Exit point format name. The exit point format name associated with the exit point.

Exit point name. The exit point name.

Exit point text description. The text for the exit point description. This field will contain blanks when a message file and message identifier are provided for the exit point description.

Exit program data. The data that is associated with the exit program.

Exit program data CCSID. The coded character set identifier (CCSID) that is used in working with the exit program data.

Exit program description indicator. Whether the exit program description is contained in a message file or text. The possible values follow:

0 The exit program description is contained in a message file.
1 The exit program description is text.

Exit program description message file name. The name of the message file that contains the exit program description. This field will contain blanks when a text description is provided for the exit program description.

Exit program description message file library name. The name of the library in which the exit program description message file resides. This field will contain blanks when a text description is provided for the exit program description.

Exit program description message ID. The message identifier for the exit program description. This field will contain blanks when a text description is provided for the exit program description.

Exit program library name. The library in which the exit program resides.

Exit program name. The name of the exit program.

Exit program number. The exit program number associated with the exit program. This number determines the processing sequence of the exit programs associated with the exit point, where the lowest number should be processed first.

Exit program text description. The text for the exit program description. This field will contain blanks when a message file and message identifier are provided for the exit program description.

Length of exit point entry. The length of an exit point entry that is returned. This value should be used in determining the offset to the next exit point entry.

Length of exit program data. The length of the exit program data that is returned.

Length of exit program entry. The length of an exit program entry, not including the exit program data, that is returned.

Maximum number of exit programs. The maximum number of exit programs that the exit point allows.

Multithreaded job action. The action to take when calling an exit program in a multithreaded job. The possible values follow:

1 Run the exit program in the current multithreaded job.

2 Run the exit program in the current multithreaded job, but send an informational message. CPI3C80 can be used as the informational message.

3 Do not run the exit program in the current multithreaded job. Depending on the exit point, do one of the following:

1. Send an escape message and do not call the exit program. CPF3C80 can be used as the escape message.
2. Send an informational message and do not call the exit program. CPF3C80 can be used as the informational message.
3. Call the exit program in a non-multithreaded job.

Number of exit point entries returned. The number of exit point entries returned. If the receiver variable is not large enough to hold all of the information, this number contains only the number of exit point entries actually returned.

Number of exit program entries returned. The number of exit program entries returned. If the receiver variable is not large enough to hold all of the information, this number contains only the number of exit program entries actually returned.

Offset to exit program data. The offset to the exit program data. The offset is from the beginning of the structure.

Offset to first exit point entry. The offset to the first exit point entry returned. The offset is from the beginning of the structure. If no entries are returned, the offset is set to zero.

Offset to first exit program entry. The offset to the first exit program entry returned. The offset is from the beginning of the structure. If no entries are returned, the offset is set to zero.

Offset to next exit program entry. The offset to the next exit program entry returned. The offset is from the beginning of the structure. If there are no more exit program entries, this value is zero.

Preprocessing exit program format name for adding an exit program. The format name for the Preprocessing Exit Program for Add.

Preprocessing exit program format name for removing an exit program. The format name for the Preprocessing Exit Program for Remove.

Preprocessing exit program format name for retrieving an exit program. The format name for the Preprocessing Exit Program for Retrieve.

Preprocessing exit program library name for adding an exit program. The library in which the Preprocessing Exit Program for Add resides.

Preprocessing exit program library name for removing an exit program. The library in which the Preprocessing Exit Program for Remove resides.

Preprocessing exit program library name for retrieving an exit program. The library in which the Preprocessing Exit Program for Retrieve resides.

Preprocessing exit program name for adding an exit program. The preprocessing exit program name that is called by the registration facility when the Add Exit Program API is called for the exit point.

Preprocessing exit program name for removing an exit program. The preprocessing exit program name that is called by the registration facility when the Remove Exit Program API is called for the exit point.

Preprocessing exit program name for retrieving exit information. The preprocessing exit program name that is called by the registration facility when the Retrieve Exit Information API is called for the exit point.

QMLTTHDACN system value. A flag that indicates whether the QMLTTHDACN system value was used in determining the multithreaded job action.

0 The QMLTTHDACN system value was not used to determine the multithreaded job action.
1 The QMLTTHDACN system value was used to determine the multithreaded job action.

Registered exit point. Whether the exit point is registered or unregistered. The possible values follow:

0 The exit point is unregistered.
1 The exit point is registered.

Reserved. An ignored field.

Threadsafe. The thread safety status of the exit program entry. The possible values follow:

0 The exit program entry is not threadsafe.
1 The threadsafe status of the exit program entry is not known.
2 The exit program entry is threadsafe.

Format for Exit Program Selection Criteria

This table shows the format for the exit program selection criteria parameter. For a detailed description of each field, see "Field Descriptions" on page 25.

Type	Field
BINARY(4)	Size of criteria entry
BINARY(4)	Comparison operator
BINARY(4)	Start position in exit program data
BINARY(4)	Length of comparison data
CHAR(*)	Comparison data

Field Descriptions

Comparison data. The data to compare to the exit program data.

Comparison operator. The comparison value to be used when comparing the exit program data with the comparison data. The following value can be specified:

1 The comparison data equals the exit program data.

Length of comparison data. The length of the data to compare to the exit program data. The length of the comparison data must be between 1 and 256.

Size of criteria entry. The size of the selection criteria entry, including this field.

Start position in exit program data. The starting position of the exit program data against which the comparison data is matched. The starting position is based on 0. Valid starting positions are from 0 through 2047.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CD2 E	Exit point name &1 not valid.
CPF3CD3 E	Exit point format name &1 not valid.
CPF3CD9 E	Requested function cannot be performed at this time.
CPF3CDA E	Registration facility repository not available for use.
CPF3CDB E	Exit point &1 with format &2 does not exist.
CPF3CE1 E	Exit program number &1 not valid.
CPF3CE2 E	Continuation handle not valid
CPF3CE3 E	Continuation handle no longer valid.
CPF3CE4 E	Comparison operator &1 not valid for exit program selection criteria.
CPF3CE6 E	Search criteria start position and length exceed boundary.
CPF3CE7 E	Number of selection criteria entries not valid.
CPF3CE8 E	Start position not valid.
CPF3CE9 E	Length of comparison data not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9802 E	Not authorized to object &2 in &3.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

"Retrieve Exit Information (QUSRTVEI, QusRetrieveExitInformation) API" on page 15 | "Registration Facility APIs," on page 1 | APIs by category

Exit Programs

These are the Exit Programs for this category.

Preprocessing Exit Program for Add

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Exit point format name	Input	Char(8)
3	Exit program number	Input	Binary(4)
4	Qualified exit program name	Input	Char(20)
5	Exit program data	Input	Char(*)
6	Length of exit program data	Input	Binary(4)
7	Exit program attributes	Input	Char(*)
8	Return code	Output	Binary(4)

The Preprocessing Exit Program for Add allows for processing to take place before an exit program is added to an exit point. The preprocessing exit program will notify the registration facility through the return code parameter whether or not to add the exit program to the exit point.

Authorities and Locks

None.

Required Parameter Group

Exit point name
INPUT; CHAR(20)

The name of the exit point to which the exit program is being added.

Exit point format name
INPUT; CHAR(8)

The format name of the exit point to which the exit program is being added.

Exit program number
INPUT; BINARY(4)

The order in which the exit programs are to be run when multiple exit programs are associated with the exit point. The valid range is 1 through 2 147 483 647, where the processing sequence is from the lowest number to the highest number. Exit program numbers do not need to be consecutive. The following special values are allowed:

-1 The next lowest available number for that specific exit point will be assigned
-2 The highest available number for that specific exit point will be assigned

When the exit point provider stores the exit program information and one of the above special values is specified, the exit point provider will assign the exit program number. Otherwise, the registration facility will assign the exit program number.

Qualified exit program name
INPUT; CHAR(20)

The exit program that is to be added, and the library in which it is located. The first 10 characters contain the exit program name, and the second 10 characters contain the name of the library in which the exit program resides. A specific library name must be specified. The special values *LIBL and *CURLIB are not supported.

Exit program data
INPUT; CHAR(*)

The exit program data supplied for the exit program that is requesting to be added to the exit point. Pointer data will not be preserved in the exit program data parameter.

Length of exit program data
INPUT; BINARY(4)

The length of the exit program data. The valid length is 0 through 2048.

Exit program attributes
INPUT; CHAR(*)

The specified information for the exit program. See "Exit Program Attribute Keys" on page 4 for more information. Any field not specified will be given the default value. The information is in the following format:

Number of variable length records
BINARY(4)

The total number of all of the variable length records.

Variable length records
The exit program attributes and their values. See "Format for Variable Length Record" on page 3 for more information.

Return code
OUTPUT; BINARY(4)

Return code to notify success or failure. The following values are allowed:

0 The registration facility should not add the exit program to the exit point and should return an error to the caller of the Add Exit Program API.

1 The registration facility should add the exit program to the exit point. If the exit point provider has specified a Preprocessing Exit Program for Retrieve and returns this return code, an error will be issued to the caller of the Add Exit Program API.

2 The registration facility will not store the exit program information. The exit point provider stored the information. If the exit point provider has not specified a Preprocessing Exit Program for Retrieve and returns this return code, an error will be issued to the caller of the Add Exit Program API.

3 The registration facility will not replace the exit program. The exit point provider replaced the exit program. If the exit point provider did not specify a Preprocessing Exit Program for Retrieve and returns this return code, an error is issued to the caller of the Add Exit Program API.

Error Messages

Error notification is done through the return code parameter. No error messages will be accepted.

Exit program introduced: V3R1

"Preprocessing Exit Program for Add" on page 26 | "Registration Facility APIs," on page 1 | APIs by category

---

Preprocessing Exit Program for Remove

Required Parameter Group:

1	Exit point name	Input	Char(20)
2	Exit point format name	Input	Char(8)
3	Exit program number	Input	Binary(4)

4	Return code	Output	Binary(4)

The Preprocessing Exit Program for Remove allows for processing to take place before an exit program is removed from an exit point. The preprocessing exit program will notify the registration facility through the return code parameter whether or not to remove the exit program from the exit point.

Authorities and Locks

None.

Required Parameter Group

Exit point name
INPUT; CHAR(20)

The exit point name from which the exit program is being removed.

Exit point format name
INPUT; CHAR(8)

The exit point format name from which the exit program is being removed.

Exit program number
INPUT; BINARY(4)

The exit program number of the exit program being removed. The following values are allowed:

-1	All exit programs associated with the specified exit point name and exit point format name will be removed.
exit program number	Only the exit program with the specified exit program number, exit point name, and exit point format name will be removed.

Return code
OUTPUT; BINARY(4)

A return code to notify success or failure. The following values are allowed:

0 The registration facility should not remove the exit program from the exit point and should return an error to the caller of the Remove Exit Program API.

1 The registration facility should remove the exit program from the exit point. If the exit point provider has specified a Preprocessing Exit Program for Retrieve and returns this return code, an error will be issued to the caller of the Remove Exit Program API.

2 The registration will not remove the exit program information. The exit point provider removed the exit program information. If the exit point provider has not specified a Preprocessing Exit Program for Retrieve and returns this return code, an error will be issued to the caller of the Remove Exit Program API.

Error Messages

Error notification is done through the return code parameter. No error messages will be accepted.

Exit program introduced: V3R1

"Preprocessing Exit Program for Remove" on page 27 | "Registration Facility APIs," on page 1 | APIs by category

Preprocessing Exit Program for Retrieve

Required Parameter Group:

1	Continuation handle	Input	Char(16)
2	Receiver variable	Output	Char(*)
3	Length of receiver variable	Input	Binary(4)
4	Format name	Input	Char(8)
5	Exit point name	Input	Char(20)
6	Exit point format name	Input	Char(8)
7	Exit program number	Input	Binary(4)
8	Exit program selection criteria	Input	Char(*)
9	Return code	Output	Binary(4)

The Preprocessing Exit Program for Retrieve allows for the exit point provider to store the exit program information. The registration facility will not store the exit program information, only exit point information. The Preprocessing Exit Program for Add and the Preprocessing Exit Program for Remove are required when this preprocessing exit program is supplied. The preprocessing exit program will notify the registration facility through the return code parameter whether or not the exit information returned is complete and accurate.

Authorities and Locks

None.

Required Parameter Group

Continuation handle
INPUT; CHAR(16)

The value returned to the API in the receiver variable when partial information is returned. This parameter is used when there is more information available to return than what could fit in the receiver variable.

Receiver variable
OUTPUT; CHAR(*)

The variable in which the preprocessing exit program will return the exit information to the registration facility. This information must be returned in the format specified in the format name parameter.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable.

Format name
INPUT; CHAR(8)

The format of the exit information to be returned. One of the following format names will be specified by the Retrieve Exit Information API:

EXTI0100	Exit point information
EXTI0200	Basic exit program information
EXTI0300	Complete exit program information

See “EXTI0100 Format” on page 18, “EXTI0200 Format” on page 19, and “EXTI0300 Format” on page 20 for more information.

Exit point name
INPUT; CHAR(20)

The name of the exit point for which information is being retrieved.

Exit point format name
INPUT; CHAR(8)

The format name associated with the exit point.

Exit program number
INPUT; BINARY(4)

The number of the exit program. When format EXTI0100 is specified, this field should be ignored. The following values are allowed:

-1	All exit programs for the exit point are returned.
exit program number	The exit program with the specified exit program number is returned. The valid range is 1 through 2 147 483 647.

Exit program selection criteria
INPUT; CHAR(*)

The selection criteria to be used when selecting which exit programs associated with the exit point are to be returned. When format EXTI0100 is specified, this field should be ignored. The information is in the following format:

Number of selection criteria
BINARY(4)

The total number of selection criteria. Zero is specified if no selection criteria are specified. The maximum value for this field is 1.

Selection criteria array
CHAR(*)

The selection criteria. See "Format for Exit Program Selection Criteria" on page 24 for more information.

Return Code
OUTPUT; BINARY(4)

The return code to notify success or failure. If there is no information to return, set the number of exit programs returned field to 0 and specify success (1) for the return code. The following values are allowed:

0	The information returned to the registration facility is incomplete or inaccurate.
1	The information returned to the registration facility is complete and accurate.

Error Messages

Error notification is done through the return code parameter. No error messages will be accepted.

API introduced: V3R1

"Preprocessing Exit Program for Retrieve" on page 29 | "Registration Facility APIs," on page 1 | APIs by category

Concepts

These are the concepts for this category.

Using Registration Facility APIs and Registration Facility

Preprocessing Exit Programs

The **registration facility** is a service that provides storage and retrieval operations for i5/OS® and non-i5/OS exit points and exit programs. An **exit point** is a specific point in a system function or program where control may be passed to one or more specified exit programs. An **exit program** is a program to which control is passed from an exit point. This exit program can then supplement standard system functions in areas such as additional authorization checks, data transformations, auditing, and so on. Examples of exit programs often can be found with the exit point documentation. This **registration facility repository** allows multiple programs to associate with a given system function or application function.

An exit point can call one program, a fixed number of programs, or all programs associated with an exit point. The exit program number associated with each exit program should be used to determine the sequence in which the exit programs are run.

An exit point can be registered multiple times with the same exit point name; however, the combination of the exit point name and the exit point format name must be unique. Each exit program will be associated with a specific exit point and exit point format. The exit point format name can be used to indicate that a change occurred to the interface of the exit point. For example, this unique name (exit point and format) could be the result of a parameter change, version change, exit program data definition, and so forth. This unique name will facilitate having different exit programs run from different versions of a product for the same exit point name.

The **exit point provider** is responsible for the following:

- Defining the exit point information
- Defining the details of the exit program, such as the number of exit programs to call and what the parameters (if any) will be
- Calling the exit programs

If you intend to provide an exit point, you should become familiar with all the APIs and the preprocessing exit programs in the registration facility part before using them. The APIs and preprocessing exit programs are interdependent.

If you intend to provide an exit program, you should become familiar with the "Add Exit Program (QUSADDEP, QusAddExitProgram) API" on page 1 and "Remove Exit Program (QUSRMVEP, QusRemoveExitProgram) API" on page 14 APIs. When developing the exit program, the exit program provider is responsible for reclaiming all resources allocated by the exit program.

The registration facility gives the exit point provider the option to perform preprocessing when an operation is requested against an exit point. The exit point provider is responsible for providing the preprocessing exit program. The preprocessing exit program is called by the registration facility before the requested function is performed on the exit point. (The requested function might be an add, remove, or retrieve operation.) The preprocessing exit program notifies the registration facility if the requested function should be completed. The following restrictions apply:

- The preprocessing exit program must exist when the exit point is registered.
- The Preprocessing Exit Program for Add and the Preprocessing Exit Program for Remove are required when a Preprocessing Exit Program for Retrieve is supplied.

"Using Registration Facility APIs and Registration Facility" | "Registration Facility 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