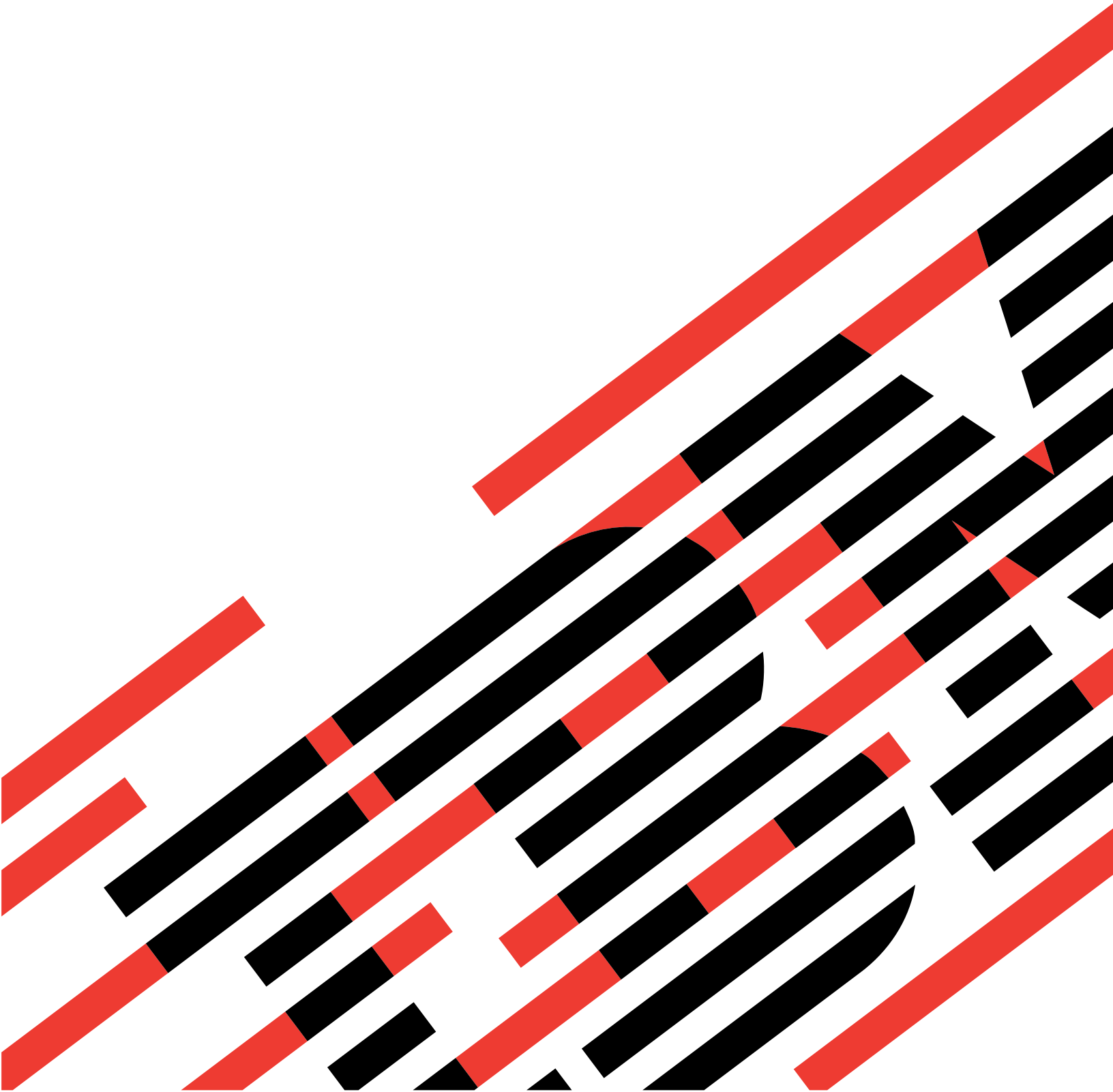
IBM

eserver

iSeries

Security -- User Function Registration APIs

Version 5 Release 3



IBM

@server

iSeries

Security -- User Function Registration APIs

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 37.

Sixth Edition (August 2005)

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

Contents

User Function Registration APIs

The user function registration APIs manage the registration and usage information for functions. To help you manage your systems, the user function registration APIs provide a mechanism for registering functions and controlling which users are allowed to use those functions. The control of user functions, however, is not a replacement for securing resources. Users who are not allowed to use a particular function are not prevented from accessing a resource through another interface.

The user function registration APIs are:

- "Change Function Usage Information (QSYCHFUI, QsyChangeFunctionUsageInfo) API" on page 2 (QSYCHFUI, QsyChangeFunctionUsageInfo) changes the usage information for a function, such as which user profiles are allowed to use a function.
- "Check User Function Usage (QSYCKUFU, QsyCheckUserFunctionUsage) API" on page 5 (QSYCKUFU, QsyCheckUserFunctionUsage) whether a user is allowed to use the specified function and returns an indication of whether the user is allowed to use the function.
- "Deregister Function (QSYDRGFN, QsyDeregisterFunction) API" on page 7 (QSYDRGFN, QsyDeregisterFunction) removes a function and all associated usage information from the registration facility.
- "Register Function (QSYRGFN, QsyRegisterFunction) API" on page 9 (QSYRGFN, QsyRegisterFunction) registers a function with the registration facility.
- "Retrieve Function Information (QSYRTVFI, QsyRetrieveFunctionInformation) API" on page 16 (QSYRTVFI, QsyRetrieveFunctionInformation) retrieves information about one or more functions.
- "Retrieve Function Usage Information (QSYRTFUI, QsyRetrieveFunctionUsageInfo) API" on page 24 (QSYRTFUI, QsyRetrieveFunctionUsageInfo) retrieves usage information for a function.
- "Retrieve User Function Information (QSYRTUFI, QsyRetrieveUserFunctionInfo) API" on page 27 (QSYRTUFI, QsyRetrieveUserFunctionInfo) retrieves usage settings for a specified user profile for one or more functions.

Top | Security APIs | APIs by category

APIs

These are the APIs for this category.

Change Function Usage Information (QSYCHFUI, QsyChangeFunctionUsageInfo) API

Required Parameter Group for QSYCHFUI:

1	Function ID
Input	Char(30)
2	Format name
Input	Char(8)
3	Function usage information
Input	Char(*)
4	Length of function usage information
Input	Binary(4)
5	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyChangeFunctionUsageInfo:

```
#include <qsyfnusg.h>

void QsyChangeFunctionUsageInfo
       (char     Function_ID[30],
        char     Format_name[8],
        void    *Function_usage_information
        int     *Length_of_function_usage_information,
        void    *Error_code);
```

Service Program: QSYFNUSG

Default Public Authority: *USE

Threadsafe: Yes

The Change Function Usage Information (OPM, QSYCHFUI; ILE, QsyChangeFunctionUsageInfo) API changes the usage information for a function. The usage information for a function indicates which user profiles are allowed or not allowed to use a function.

The usage information is stored with the user profile. To save and restore the usage information, you must use the same methods as with other user profile information (Save Security Data (SAVSECDTA) command, Restore User Profiles (RSTUSRPRF) command, Restore Authority (RSTAUT) command).

Authorities and Locks

API Public Authority
*USE

Authority Required
*SECADM special authority

Usage Information Lock
*EXCL

Required Parameter Group

Function ID
INPUT; CHAR(30)

The ID of the function for which usage information is being changed.

Format name
INPUT; CHAR(8)

The format of the function usage information.

The valid value is:

"CFUI0100 Format"	Function usage information

Function usage information
INPUT; CHAR(*)

The usage information that is being changed for the specified function. See "CFUI0100 Format" for the definition of the fields for this parameter.

Length of function usage information
INPUT; BINARY(4)

The length of the function usage information. This area must be as large as the format specified.

Error code
I/O; CHAR(*)

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

CFUI0100 Format

The following table shows the information that must be specified in the function usage information parameter when format CFUI0100 is specified. For a detailed description of each field, see "Field Descriptions" on page 4.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of usage information entries
Usage information entry. These fields are repeated for each usage information entry returned.			
		CHAR(10)	User profile name
		CHAR(1)	Usage setting

Field Descriptions

Number of usage information entries. The total number of usage information entries. This value must be greater than 0.

Usage information entries. The entries that contain the usage information.

Usage setting. Whether the user is allowed to use the function or not.

The following values can be specified:

0	The user's previous setting is removed.
1	The user is not allowed to use the function.
2	The user is allowed to use the function.

User profile name. The name of the user profile whose usage setting is being changed. The user profile must exist.

Error Messages

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF222E E	&1 special authority is required.
CPF228A E	Function &1 not registered.
CPF229B E	Operation not allowed for function &1.
CPF229C E	Not all usage information changed for function &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
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.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9810 E	Library &1 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Top | Security APIs | APIs by category

Check User Function Usage (QSYCKUFU, QsyCheckUserFunctionUsage) API

Required Parameter Group for QSYCKUFU:

1	Usage indicator
Output	Char(1)
2	Function ID
Input	Char(30)
3	User profile name
Input	Char(10)
4	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyCheckUserFunctionUsage:

```
#include <qsyfnusg.h>

void QsyCheckUserFunctionUsage
       (char    *Usage_indicator,
        char     Function_ID[30],
        char     User_profile_name[10],
        void    *Error_code);
```

Service Program: QSYFNUSG

Default Public Authority: *USE

Threadsafe: Yes

The Check User Function Usage (OPM, QSYCKUFU; ILE, QsyCheckUserFunctionUsage) API checks to see if a user is allowed to use the specified function, and returns an indication of whether the user is allowed to use the function. When the check is made, the usage setting for the user, its group, the default usage value, and the allow *ALLOBJ indicator for the function are taken into account. Following are the steps the system takes to determine the usage indicator for the user to the function:

1. Is the *ALLOBJ indicator for the function set to 1 and does the user have *ALLOBJ special authority? If yes to both, set the returned usage indicator to usage allowed and return. Otherwise, continue with step 2.
2. Does the user have a usage setting for the function? If yes, set the returned usage indicator to the usage indicator for the user and return. Otherwise, continue with step 3.
3. Does the user have any groups? If yes, continue with step 3a. Otherwise, go to step 4.
   a. Repeat the following steps for each group:

1) Is the *ALLOBJ indicator for the function set to 1 and does the group have *ALLOBJ special authority? If yes to both, set the returned usage indicator to usage allowed and return. Otherwise, go to step 3a2.
2) Does the group have a usage setting for the function? If yes, go to step 3a2a. Otherwise, go look at the next group, starting over with step 3a1.
   a) Does the group's usage setting allow usage of the function? If yes, set the returned usage indicator to usage allowed and return. Otherwise, set a flag that indicates a group usage setting is found and go look at the next group, starting over with step 3a1.

b. Is the group usage setting found flag set on? If yes, set the returned usage indicator to usage not allowed and return. Otherwise, continue with step 4.

4. Set the returned usage indicator to the default usage value and return.

Authorities and Locks

API Public Authority
*USE

Function Registration Lock
*SHRNUP

Required Parameter Group

Usage indicator
OUTPUT; CHAR(1)

Whether the user is allowed to use the specified function.
This parameter contains one of the following values:

1	The user is not allowed to use the specified function.
2	The user is allowed to use the specified function.

Function ID
INPUT; CHAR(30)

The ID of the function to check usage information for.

User profile name
INPUT; CHAR(10)

The name of the user to check for usage to the specified function.

You can specify the following special value:

**CURRENT*	The usage check is made for the user currently running.

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
CPF2204 E	User profile &1 not found.
CPF2225 E	Not able to allocate internal system object.
CPF228A E	Function &1 not registered.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.

Message ID	Error Message Text
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.
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.
CPF9810 E	Library &1 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Deregister Function (QSYDRGFN, QsyDeregisterFunction) API

Required Parameter Group for QSYDRGFN:

1	Function ID
Input	Char(30)
2	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyDeregisterFunction:

```
#include <qsyrgfn1.h>

void QsyDeregisterFunction
       (char     Function_ID[30],
        void    *Error_code);
```

Service Program: QSYRGFN1

Default Public Authority: *USE

Threadsafe: Yes

The Deregister Function (OPM, QSYDRGFN; ILE, QsyDeregisterFunction) API removes a function and all associated usage information from the registration facility.

Authorities and Locks

API Public Authority
*USE

Authority Required
*SECADM special authority

Function Registration Lock
*EXCL

Required Parameter Group

Function ID
INPUT; CHAR(30)

The function ID for the function being removed.
The following can be specified for the function ID:

*generic**	All functions IDs that have IDs beginning with the generic string.
function ID	Specific function ID.

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
CPF222E E	&1 special authority is required.
CPF228A E	Function &1 not registered.
CPF228B E	Function &1 cannot be removed from the registry.
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.
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.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Top | Security APIs | APIs by category

Register Function (QSYRGFN, QsyRegisterFunction) API

Required Parameter Group for QSYRGFN:

1	Function ID
Input	Char(30)
2	Function controls
Input	Char(*)
3	Error Code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyRegisterFunction:

```
#include <qsyrgfn1.h>

void QsyRegisterFunction
       (char                  Function_ID[30],
        Qsy_Func_Controls_T  *Function_controls,
        void                 *Error_code);
```

Service Program: QSYRGFN1

Default Public Authority: *USE

Threadsafe: Yes

The Register Function (OPM, QSYRGFN; ILE, QsyRegisterFunction) API registers a function with the registration facility. The function controls provide information to help manage and control the use of the function.

You can update a function entry by reregistering the function (using the replace control key) with new values for the function control keys. The registration facility will update the control keys and maintain the current list of usage settings that are associated with the function. The following conditions apply to updating the function control keys:

- Function category: This control key is set the first time the function is registered and cannot be changed.
- Function type: This control key is set the first time the function is registered and cannot be changed.

When a function is registered, the registration information is stored using the Exit Point Registration Facility. Each function is registered as an exit program within an exit point. The exit point that the function is registered in depends on the category of the function:

- Locally managed client functions within iSeries Navigator are registered in the QIBM_QSY_OPNAVCLIENT exit point.
- Locally managed client functions that are not within iSeries Navigator are registered in the QIBM_QSY_OTHERCLIENT exit point.
- Host functions are registered in the QIBM_QSY_HOSTFUNC exit point.
- Centrally managed client functions within iSeries Navigator are registered in the QIBM_QSY_OPNAVCENTRL exit point.
- Centrally managed client functions that are not within iSeries Navigator are registered in the QIBM_QSY_OTHERCENTRL exit point.

The Exit Point Registration Facility is used only for storing the function registration information. The exit programs within these exit points are never called, so the formats associated with these exit points (FCNR0100 and FCNR0200) are never used or documented.

Note: You must use this API and the Deregister Function (QSYDRGFN, QsyDeregisterFunction) API to manage the function registration entries. If the Exit Point Registration Facility APIs are used, the results will be unpredictable.

Authorities and Locks

API Public Authority
*USE

Authority Required
*SECADM special authority

Function Registration Lock
*EXCL

Required Parameter Group

Function ID
INPUT; CHAR(30)

The function ID to register. IBM functions are named QIBM_Q*ccc_name*, where _Q*ccc* is the componenet qualifier and *ccc* is the component identifier, or are named QIBM_*wccc_name*, where *_wccc* is the component qualifier, *w* is a character A through I, and *ccc* is the component identifier. The component qualifier is included in IBM function IDs if the function is associated with a specific component. Otherwise, IBM function IDs will not contain the component qualifier. User-supplied function IDs should not preface their function ID with QIBM. User-supplied function IDs should start with the company name to eliminate most problems that involve unique names.

The first character of the function ID must be one of the following:

A-Z	Uppercase A-Z

The remaining characters in the function ID must be made up of the following characters:

A-Z	Uppercase A-Z
0-9	Digits 0-9
.	Period
_	Underscore

Function controls
INPUT; CHAR(*)

The function control fields for managing the function. Any field not specified will be given the default value. Refer to “Function Control Keys” on page 12 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 function 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.”

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Length of variable length record
4	4	BINARY(4)	Function 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 function control is to be set.

Function control key. The function control to be set. Refer to “Function Control Keys” on page 12 for more information.

Length of data. The length of the function control value.

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

Function Control Keys

The following table shows the valid function 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)	Function category
2	CHAR(1)	Function type
3	CHAR(30)	Function product ID
4	CHAR(30)	Function group ID
5	CHAR(27)	Qualified message file name and message identifier for function name
6	CHAR(132)	Function name
7	BINARY(4)	Function name CCSID
8	CHAR(27)	Qualified message file name and message identifier for function description
9	CHAR(132)	Function description
10	BINARY(4)	Function description CCSID
11	CHAR(1)	Default usage
12	CHAR(1)	*ALLOBJ indicator
13	CHAR(1)	Replace

Field Descriptions

***ALLOBJ indicator.**

Whether a user with *ALLOBJ special authority can use the function. The default value is 1. This key can be specified only if the function type control key is set to 3 (administrable function).

0 For a user with *ALLOBJ special authority to use the function, the usage information specified for the function must indicate that the user is allowed to use the function for one of the following reasons:

1. the user is allowed usage,
2. one of its groups is allowed usage, or
3. the default setting is to allow usage.

1 A user with *ALLOBJ special authority is always allowed to use the function.

Default usage. The default usage for the function. The default usage is used if the user or one of its groups does not have a specific usage setting. This key can be specified only if the function type control key is set to 3 (administrable function). The default value is 2.

1 The default usage is to not allow usage of the function.

2 The default usage is to allow usage of the function.

Function category. The category of the function. This control is set when the function is registered and cannot be changed. The default value is 3.

1 The function is a locally managed client function within iSeries Navigator. The function defines code that runs on the client, is locally managed, and is included in the iSeries Navigator tree structure.

2 The function is a locally managed client function, not within iSeries Navigator. The function defines code that runs on the client and is locally managed, but is not included in the iSeries Navigator tree structure.

3 The function is a host function. The function defines code that runs on the server.

4 The function is a centrally managed client function within iSeries Navigator. The function defines code that runs on the client, is centrally managed, and is included in the iSeries Navigator tree structure.

5 The function is a centrally managed client function, not within iSeries Navigator. The function defines code that runs on the client and is centrally managed, but is not included in the iSeries Navigator tree structure.

Only client based applications should use category 1, 2, 4, or 5. See Application Administration Concepts in the iSeries Navigator topic for more information on the difference between locally managed and centrally managed client functions.

Function description. The text for the function description. The default value is blanks.

Function description CCSID. The CCSID value associated with the function description. The default value is 0.

0 The current job default CCSID value is associated with the function description.

CCSID The CCSID value associated with the function description. The CCSID value must be from 1 to 65535. If 65535 is specified, then the data will not be converted to the desired CCSID when the function is retrieved.

Function group ID. The ID of the function group with which this function will be grouped. The function group must exist, have a function type of function group (2), and have the same category as the function being registered. If a value of *NONE is specified, then the function is not grouped under a function group. The default value is *NONE. This key cannot be specified if the function type control key is set to 1 (function product).

Function name. The text name for the function ID. The function name will be used to identify the function when using the iSeries Navigator Application Administration support. If this control is not specified, the function ID will be used to identify the function.

Function name CCSID. The CCSID value associated with the function name. The default value is 0.

0 The current job default CCSID value is associated with the function name.

CCSID The CCSID value associated with the function name. The CCSID value must be from 1 to 65535. If 65535 is specified, then the data will not be converted to the desired CCSID when the function is retrieved.

Function product ID. The ID of the product for which the function is being registered. This key cannot be specified if the function type control key is set to 1 (function product). This key must be specified if the function type control key is set to 2 (function group) or 3 (administrable function).

Function type. The type of function. This control is set when the function is registered and cannot be changed. The default value is 3.

1 The function is a function product. No usage information is stored for function products. The purpose is to group functions under a product when the functions are being administered by the iSeries Navigator Application Administration support.

2 The function is a function group. No usage information is stored for function groups. The purpose is to group functions within a product when the functions are being administered by the iSeries Navigator Application Administration support.

3 The function is an administrable function. Usage information can be stored and checked for this function.

Qualified message file name and message identifier for function description. A message file and message identifier that contains the function description. 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 name and message identifier for function name. A message file and message identifier that contains the function name. 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.

Replace. Whether to replace an existing registered function. The default value is 0.

0 Do not replace an existing registered function. If this value is specified and the function is already registered, the request will fail.

1 Replace an existing registered function. If this value is specified and the function is not already registered, the function will be registered. If the function is already registered, only the function control keys that are specified on this call are replaced. Any other function control keys that were previously specified will keep their values.

» 2 Replace an existing registered function, but do not replace function control keys that are controlled by a system administrator. If this value is specified and the function is not already registered, the function will be registered. If the function is already registered, only the function control keys that are specified on this call are replaced. Any other function control keys that were previously specified will keep their values. Function control keys that are controlled by a system administrator are not replaced, even if they are specified on this call. These function control keys include:

- *ALLOBJ indicator
- Default usage

This value should be used by install exit programs to ensure that values set by the system adminstrator are not replaced by the install exit program. «

Qualified Message File Format

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

Offset		Type	Field
Dec	Hex		
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. No special values are supported.

Message file name. The name of the message file that contains the function name or function description.

Message identifier. The message identifier for the function name or function description.

Error Messages

Message ID	Error Message Text
CPFA0AA E	Error occurred while attempting to obtain space.
CPF2225 E	Not able to allocate internal system object.
CPF222E E	&1 special authority is required.
CPF228C E	Function ID &1 not valid.
CPF228D E	Function group &1 not registered in same category.
CPF228E E	Function product &1 not registered in same category.
CPF228F E	Function &1 already registered.
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.
CPF3C88 E	Number of variable length records &1 is 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.
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.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Top | Security APIs | APIs by category

Retrieve Function Information (QSYRTVFI, QsyRetrieveFunctionInformation) API

Required Parameter Group for QSYRTVFI:

1	Continuation handle
Input	Char(20)
2	Receiver variable
Output	Char(*)
3	Length of receiver variable
Input	Binary(4)
4	Format name
Input	Char(8)
5	Function selection criteria
Input	Char(*)
6	Desired CCSID
Input	Binary(4)
7	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyRetrieveFunctionInformation:

```
#include <qsyfnusg.h>

void QsyRetrieveFunctionInformation
       (char            *Continuation_handle[20],
        void            *Receiver_variable,
        int             *Length_of_receiver_variable,
        char            *Format_name[8],
        Qsy_Selcrtr_T   *Function_selection_criteria,
        int             *Desired_CCSID,
        void            *Error_code);
```

Service Program: QSYFNUSG

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Function Information (OPM, QSYRTVFI; ILE, QsyRetrieveFunctionInformation) API retrieves information about one or more functions.

Authorities and Locks

API Public Authority
*USE

Function Registration Lock
*SHRNUP

Required Parameter Group

Continuation handle
INPUT; CHAR(20)

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 function 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 function information to be returned.

You must use the following format name:

"FCNI0100 Format" on page 18	Function information

Function selection criteria
INPUT; CHAR(*)

The selection criteria to be used when selecting which functions are returned. No CCSID normalization is performed. It is recommended that you use characters from the invariant character set for the comparison data.

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 Function Selection Criteria" on page 21 for more information.

Desired CCSID
INPUT; BINARY(4)

The CCSID the returned text fields should be converted to. The text fields will be returned in this CCSID even if data loss occurs. If you want to ensure that data loss does not occur, you may specify 65535 or 13488 (UCS-2).

The following can be specified for the desired CCSID:

0 The text fields will be converted to the default CCSID for the job.

CCSID The text fields will be converted to the specified CCSID. The CCSID value must be from 1 to 65535. If 65535 is specified, then no CCSID conversion will be done on the text.

Error code

I/O; CHAR(*)

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

FCNI0100 Format

The following information is returned for the FCNI0100 format. This format provides information on a function. For a detailed description of each field, see "Field Descriptions" on page 19.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(20)	Continuation handle
28	1C	BINARY(4)	Offset to first function entry
32	20	BINARY(4)	Number of function entries returned
36	24	BINARY(4)	Length of function entry
40	28	CHAR(*)	Reserved
Function entry information. These fields are repeated for each function entry returned.			
		CHAR(30)	Function ID
		CHAR(1)	Function category
		CHAR(1)	Function type
		CHAR(10)	Function-name message-file name
		CHAR(10)	Function-name message-file library name
		CHAR(7)	Function-name message ID
		CHAR(330)	Function-name message text
		CHAR(3)	Reserved
		BINARY(4)	Function-name message-text CCSID
		CHAR(330)	Function name
		CHAR(2)	Reserved
		BINARY(4)	Function name CCSID
		CHAR(10)	Function-description message-file name
		CHAR(10)	Function-description message-file library name
		CHAR(7)	Function-description message ID
		CHAR(330)	Function-description message text
		CHAR(3)	Reserved

Offset		Type	Field
Dec	Hex		
		BINARY(4)	Function-description message text CCSID
		CHAR(330)	Function description
		CHAR(2)	Reserved
		BINARY(4)	Function description CCSID
		CHAR(30)	Function product ID
		CHAR(30)	Function group ID
		CHAR(1)	Default usage
		CHAR(1)	*ALLOBJ indicator
		CHAR(1)	Usage information indicator
		CHAR(*)	Reserved

Field Descriptions

***ALLOBJ indicator.** Whether a user with *ALLOBJ special authority can use the function. If this is not an administrable function, then this field is blank.

The possible values follow:

0 The user, its groups, or default must allow usage of the function.
1 A user with *ALLOBJ special authority is always allowed to use the function.

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.

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.

Default usage. The default usage for the function. The default usage is used if the user or one of its groups does not have a specific usage setting. If this is not an administrable function, then this field is blank.

The possible values follow:

1 The default usage does not allow usage of the function.
2 The default usage allows usage of the function.

Function category. Whether the function is a client or host function.

The possible values follow:

1 The function is a locally managed client function within iSeries Navigator.
2 The function is a locally managed client function, not within iSeries Navigator.
3 The function is a host function.
4 The function is a centrally managed client function within iSeries Navigator.
5 The function is a centrally managed client function, not within iSeries Navigator.

Function description. The text for the function description. This value is converted to the CCSID value specified in the desired CCSID parameter.

Function description CCSID. The CCSID value that is associated with the function description in the registration facility.

Function-description message-file name. The name of the message file that contains the function description. If no message file name is specified, this field contains blanks.

Function-description message-file library name. The name of the library in which the function description message file resides. If no message file name is specified, this field contains blanks.

Function-description message ID. The message identifier for the function description. If no message file name is specified, this field contains blanks.

Function-description message text. The first-level text for the function-description message ID. This value is converted to the CCSID value specified in the desired CCSID parameter.

When the message text is retrieved from the message file, the message file library is added to the product portion of the library list, and *LIBL is specified for the library name. If the library cannot be added to the product portion of the library list, then *LIBL is still used to search for the message. If the message is not found, then the message file library is searched for the message.

If no message file name is specified, this field contains blanks.

Function-description message-text CCSID. The CCSID value that the function-description message text is stored in.

Function group ID. The ID of the function group that the function is grouped with. If the function is not grouped with a function group, this field is set to *NONE.

Function ID. The function ID.

Function name. The text for the function name. This value is converted to the CCSID value specified in the desired CCSID parameter.

Function name CCSID. The CCSID value that is associated with the function name in the registration facility.

Function-name message-file name. The name of the message file that contains the function name. If no message file name is specified, this field contains blanks.

Function-name message-file library name. The name of the library in which the function name message file resides. If no message file name is specified, this field contains blanks.

Function-name message ID. The message identifier for the function name. If no message file name is specified, this field contains blanks.

Function-name message text. The first-level text for the function-name message ID. This value is converted to the CCSID value specified in the desired CCSID parameter.

When the message text is retrieved from the message file, the message file library is added to the product portion of the library list, and *LIBL is specified for the library name. If the library cannot be added to the product portion of the library list, then *LIBL is still used to search for the message. If the message is not found, then the message file library is searched for the message.

If no message file name is specified, this field contains blanks.

Function-name message-text CCSID. The CCSID value that the function-name message text is stored in.

Function product ID. The ID of the product that the function is registered for.

Function type. The type of function.

The possible values follow:

1 The function is a function product.
2 The function is a function group.
3 The function is an administrable function.

Length of function entry. The length of a function entry that is returned. This value should be used in determining the displacement to the next function entry.

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

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

Reserved. An ignored field.

Usage information indicator. Whether there is usage information defined for the function. Usage information is the list of users and groups that have a specific usage setting specified for the function. Usage information is set using the Change Function Usage Information (OPM, QSYCHFUI; ILE, QsyChangeFunctionUsageInfo) API. If this is not an administrable function, then this field is blank.

The possible values follow:

0 There is no usage information defined for the function.
1 There is usage information defined for the function.

Format for Function Selection Criteria

This table shows the format for the function selection criteria parameter. For a detailed description of each field, see "Field Descriptions" on page 22.

Type	Field
BINARY(4)	Size of criteria entry
BINARY(4)	Comparison operator
BINARY(4)	Function control key

Type	Field
BINARY(4)	Length of comparison data
CHAR(*)	Comparison data

Field Descriptions

Comparison data. The data to compare to the function information.

Comparison operator. The comparison value to be used when comparing the function information with the comparison data.

The following value can be specified:

1 The comparison data equals the function information

Function control key. The function control to be compared. Refer to "Function Control Keys" for more information.

Length of comparison data. The length of the data to compare to the function information. The length of the comparison data must be valid for the function control key that is specified.

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

Function Control Keys

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

Key	Type	Field
1	CHAR(1)	Function category
2	CHAR(1)	Function type
3	CHAR(30)	Function product ID
4	CHAR(30)	Function group ID
5	CHAR(30)	Function ID

Field Descriptions

Function category. The category of the function.

The possible values are:

1 Locally managed client functions within iSeries Navigator are selected.
2 Locally managed client functions not within iSeries Navigator are selected.
3 Host functions are selected.
4 Centrally managed client functions within iSeries Navigator are selected.
5 Centrally managed client functions not within iSeries Navigator are selected.
7 All locally managed client functions are selected. This includes all functions in categories 1 and 2.
8 All centrally managed client functions are selected. This includes all functions in categories 4 and 5.
9 All client functions are selected. This includes all functions in categories 1, 2, 4, and 5.

Function group ID. All functions that have this function group are selected. The special value of *NONE can be specified to select functions that do not have a function group specified.

Function ID. The name of the functions to select.

The following can be specified for the function ID:

*generic**	All function IDs that begin with the generic string are selected.
function ID	The specific function ID is selected.

Function product ID. All functions that have this function product are selected.

Function type. The type of function.

The possible values are:

1	Function products are selected.
2	Function groups are selected.
3	Administrable functions are selected.

Error Messages

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF3C19 E	Error occurred with receiver variable specified.
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.
CPF3C81 E	Value for key &1 not valid.
CPF3C82 E	Key &1 not valid for API &2.
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.
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.
CPF3CE7 E	Number of selection criteria entries 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.
CPF9810 E	Library &1 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Top | Security APIs | APIs by category

Retrieve Function Usage Information (QSYRTFUI, QsyRetrieveFunctionUsageInfo) API

Required Parameter Group for QSYRTFUI:

1	Receiver variable
Output	Char(*)
2	Length of receiver variable
Input	Binary(4)
3	Format name
Input	Char(8)
4	Function ID
Input	Char(30)
5	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyRetrieveFunctionUsageInfo:

```
#include <qsyfnusg.h>

void QsyRetrieveFunctionUsageInfo
       (void    *Receiver_variable,
        int     *Length_of_receiver_variable,
        char     Format_name[8],
        char     Function_ID[30],
        void    *Error_code);
```

Service Program: QSYFNUSG

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Function Usage Information (OPM, QSYRTFUI; ILE, QsyRetrieveFunctionUsageInfo) API retrieves usage information for a function.

Authorities and Locks

API Public Authority
*USE

Authority Required
*SECADM special authority

Usage Information Lock
*SHRNUP

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The variable that is to receive the function usage information.

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 function usage information to be returned.

You must use the following format name:

FNUI0100 Function usage information

Refer to "FNUI0100 Format" for more information.

Function ID
INPUT; CHAR(30)

The ID of the function for which usage information is being retrieved.

Error code
I/O; CHAR(*)

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

FNUI0100 Format

The following information is returned for the FNUI0100 format. This format provides usage information for a function. For a detailed description of each field, see "Field Descriptions" on page 26.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Offset to first function usage entry
12	C	BINARY(4)	Number of function usage entries returned
16	10	BINARY(4)	Length of function usage entry
20	14	CHAR(*)	Reserved
Function usage entry information. These fields are repeated for each function usage entry returned.			
		CHAR(10)	User profile name
		CHAR(1)	Usage setting
		CHAR(1)	User profile type
		CHAR(*)	Reserved

Field Descriptions

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

Bytes returned. The number of bytes of data returned.

Length of function usage entry. The length of a function usage entry that is returned. This value should be used in determining the displacement to the next function usage entry.

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

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

Reserved. An ignored field.

Usage setting. Whether the user profile is allowed to use the function or not.

The possible values follow:

1 The user profile is not allowed to use the function.
2 The user profile is allowed to use the function.

User profile name. The name of the user profile that has a usage setting for this function.

User profile type. Whether the user profile is a user or a group.

The possible values follow:

0 The user type is unknown. This value is returned if an error occurred while determining the type of user.
1 The user profile is a user (does not have a GID value).
2 The user profile is a group (has a GID value).

Error Messages

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF222E E	&1 special authority is required.
CPF228A E	Function &1 not registered.
CPF229B E	Operation not allowed for function &1.
CPF3C19 E	Error occurred with receiver variable specified.
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.
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.
CPF9810 E	Library &1 not found.

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

API introduced: V4R3

Retrieve User Function Information (QSYRTUFI, QsyRetrieveUserFunctionInfo) API

Required Parameter Group for QSYRTUFI:

1	Continuation handle
Input	Char(20)
2	Receiver variable
Output	Char(*)
3	Length of receiver variable
Input	Binary(4)
4	Format name
Input	Char(8)
5	User profile name
Input	Char(10)
6	Function selection criteria
Input	Char(*)
7	Desired CCSID
Input	Binary(4)
8	Error code
I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

Syntax for QsyRetrieveUserFunctionInfo:

```
#include <qsyfnusg.h>

void QsyRetrieveUserFunctionInfo
       (char            *Continuation_handle*[20],
        void           **Receiver_variable*,
        int             **Length_of_receiver_variable*,
        char            *Format_name*[8],
        char            *User_profile_name*[10],
        Qsy_Selcrtr_T   **Function_selection_criteria*,
        int             **Desired_CCSID*,
        void            **Error_code*);
```

Service Program: QSYFNUSG

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve User Function Information (OPM, QSYRTUFI; ILE, QsyRetrieveUserFunctionInfo) API retrieves usage settings for a specified user profile for one or more functions.

Authorities and Locks

API Public Authority
*USE

*Authority Required (if user profile name is not *CURRENT)*
*SECADM special authority or

*READ to the user profile

Function Registration and Usage Information Lock
*SHRNUP

Required Parameter Group

Continuation handle
INPUT; CHAR(20)

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 usage 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 usage information to be returned.

You must use one of the following format names:

"UFNI0100 Format" on page 30	User function usage information
"UFNI0200 Format" on page 30	The same information as UFNI0100, plus an indicator of where the user's usage setting comes from, the function name information, the function group ID, and the function product ID.
"UFNI0300 Format" on page 31	The same information as UFNI0200, plus additional indicators of where the user's usage setting and source come from if the *ALLOBJ authority is ignored, and where the user's usage setting and source come from if the *ALLOBJ and specific usage setting are ignored.

User profile name
INPUT; CHAR(10)

The name of the user profile to retrieve function usage information for.

You may specify the following special value.

**CURRENT*	The usage information for the user currently running is returned.

Function selection criteria
INPUT; CHAR(*)

The selection criteria to be used when selecting which functions to return usage information for. No CCSID normalization is performed. It is recommended that you use characters from the invariant character set for the comparison data.

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. If 0 is specified, usage information for all registered functions will be returned. The maximum value for this field is 1.
Selection criteria array	CHAR(*) The selection criteria. Refer to "Format for Function Selection Criteria" on page 34 for more information.

Desired CCSID
INPUT; BINARY(4)

The CCSID that the returned text fields should be converted to. The text fields will be returned in this CCSID even if data loss occurs. If you want to ensure that data loss does not occur, you may specify 65535 or 13488 (UCS-2).

The following can be specified for the desired CCSID:

0	The text fields are converted to the default CCSID for the job.
CCSID	The text fields are converted to the specified CCSID. The CCSID value must be from 1 to 65535. If 65535 is specified, then no CCSID conversion is done on the text.

Error code
I/O; CHAR(*)

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

UFNI0100 Format

The following information is returned for the UFNI0100 format. This format provides function usage information for a user profile. For a detailed description of each field, see "Field Descriptions" on page 32.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(20)	Continuation handle
28	1C	BINARY(4)	Offset to first function entry
32	20	BINARY(4)	Number of function entries returned
36	24	BINARY(4)	Length of function entry
40	28	CHAR(*)	Reserved
Function entry information. These fields are repeated for each function entry returned.			
		CHAR(30)	Function ID
		CHAR(1)	Usage indicator
		CHAR(*)	Reserved

UFNI0200 Format

The following information is returned for the UFNI0200 format. This format provides function usage information for a user profile, plus the usage source indicator, function name information, function group ID, and function product ID. For a detailed description of each field, see "Field Descriptions" on page 32.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(20)	Continuation handle
28	1C	BINARY(4)	Offset to first function entry
32	20	BINARY(4)	Number of function entries returned
36	24	BINARY(4)	Length of function entry
40	28	CHAR(*)	Reserved
Function entry information. These fields are repeated for each function entry returned.			
		CHAR(30)	Function ID
		CHAR(1)	Usage indicator
		CHAR(1)	Usage source
		CHAR(10)	Function-name message-file name
		CHAR(10)	Function-name message-file library name
		CHAR(7)	Function-name message ID
		CHAR(330)	Function-name message text

Offset		Type	Field
Dec	Hex		
		CHAR(3)	Reserved
		BINARY(4)	Function-name message-text CCSID
		CHAR(330)	Function name
		CHAR(2)	Reserved
		BINARY(4)	Function name CCSID
		CHAR(30)	Function product ID
		CHAR(30)	Function group ID
		CHAR(*)	Reserved

UFNI0300 Format

The following information is returned for the UFNI0300 format. This format returns the same information as format UFNI0200, plus where the user's usage setting and usage source come from if the *ALLOBJ authority is ignored, and where the user's usage setting and usage source come from if the *ALLOBJ authority and specific usage setting are ignored. For a detailed description of each field, see "Field Descriptions" on page 32.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(20)	Continuation handle
28	1C	BINARY(4)	Offset to first function entry
32	20	BINARY(4)	Number of function entries returned
36	24	BINARY(4)	Length of function entry
40	28	CHAR(*)	Reserved
Function entry information. These fields are repeated for each function entry returned.			
		CHAR(30)	Function ID
		CHAR(1)	Usage indicator
		CHAR(1)	Usage source
		CHAR(10)	Function-name message-file name
		CHAR(10)	Function-name message-file library name
		CHAR(7)	Function-name message ID
		CHAR(330)	Function-name message text
		CHAR(3)	Reserved
		BINARY(4)	Function-name message-text CCSID
		CHAR(330)	Function name
		CHAR(2)	Reserved
		BINARY(4)	Function name CCSID
		CHAR(30)	Function product ID
		CHAR(30)	Function group ID

Offset		Type	Field
Dec	Hex		
		CHAR(1)	Usage indicator ignoring *ALLOBJ
		CHAR(1)	Usage source ignoring *ALLOBJ
		CHAR(1)	Usage indicator ignoring *ALLOBJ and ignoring specific usage setting
		CHAR(1)	Usage source ignoring *ALLOBJ and ignoring specific usage setting
		CHAR(*)	Reserved

Field Descriptions

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

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.

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.

Function group ID. The ID of the function group that the function is grouped with. If the function is not grouped with a function group, this field is set to *NONE.

Function ID. The ID of the function.

Function name. The text for the function name. This value is converted to the CCSID value specified in the desired CCSID parameter.

Function name CCSID. The CCSID value that is associated with the function name in the function registration facility.

Function-name message-file name. The name of the message file that contains the function name. If no message file name is specified, this field contains blanks.

Function-name message-file library name. The name of the library in which the function name message file resides. If no message file name is specified, this field contains blanks.

Function-name message ID. The message identifier for the function name. If no message file name is specified, this field contains blanks.

Function-name message text. The first-level text for the function-name message ID. This value is converted to the CCSID value specified in the desired CCSID parameter.

When the message text is retrieved from the message file, the message file library is added to the product portion of the library list, and *LIBL is specified for the library name. If the library cannot be added to the product portion of the library list, then *LIBL is still used to search for the message. If the message is not found, then the message file library is searched for the message.

If no message file name is specified, this field contains blanks.

Function-name message-text CCSID. The CCSID value that the function-name message text is stored in.

Function product ID. The ID of the product that the function is registered for.

Length of function entry. The length of a function entry that is returned. This value should be used in determining the displacement to the next function entry.

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

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

Reserved. An ignored field.

Usage indicator. Whether the user is allowed to use the function.

This parameter contains one of the following values:

1 The user is not allowed to use the function.
2 The user is allowed to use the function.

Usage indicator ignoring *ALLOBJ. Whether the user is allowed to use the function if this API ignores the *ALLOBJ authority.

This parameter contains one of the following values:

1 The user is not allowed to use the function.
2 The user is allowed to use the function.

Usage indicator ignoring *ALLOBJ and ignoring specific usage setting. Whether the user is allowed to use the function if this API ignores the *ALLOBJ authority and his specific usage setting.

This parameter contains one of the following values:

1 The user is not allowed to use the function.
2 The user is allowed to use the function.

Usage source. An indicator as to why the user is allowed or is not allowed to use the function.

This parameter contains one of the following values:

1 The user has *ALLOBJ special authority.
2 The user has a specific usage setting for the function.
3 One of the user's groups has *ALLOBJ special authority.
4 One of the user's groups has a specific usage setting for the function.
5 The default usage setting for the function was used.
9 The usage source could not be determined.

Usage source ignoring *ALLOBJ. An indicator as to why the user is allowed or is not allowed to use the function if this API ignores the *ALLOBJ authority.

This parameter contains one of the following values:

2 The user has a specific usage setting for the function.
3 One of the user's groups has *ALLOBJ special authority.
4 One of the user's groups has a specific usage setting for the function.
5 The default usage setting for the function was used.
9 The usage source could not be determined.

Usage source ignoring *ALLOBJ and ignoring specific usage setting. An indicator as to why the user is allowed or is not allowed to use the function if this API ignores the *ALLOBJ authority and specific usage setting.

This parameter contains one of the following values:

3 One of the user's groups has *ALLOBJ special authority.
4 One of the user's groups has a specific usage setting for the function.
5 The default usage setting for the function was used.
9 The usage source could not be determined.

Format for Function Selection Criteria

This table shows the format for the function selection criteria parameter. For a detailed description of each field, see "Field Descriptions."

Type	Field
BINARY(4)	Size of criteria entry
BINARY(4)	Comparison operator
BINARY(4)	Function control key
BINARY(4)	Length of comparison data
CHAR(*)	Comparison data

Field Descriptions

Comparison data. The data to compare to the function information.

Comparison operator. The comparison value to be used when comparing the function information with the comparison data.

The following value can be specified:

1 The comparison data equals the function information

Function control key. The function control to be compared. Refer to "Function Control Keys" on page 35 for more informaiton.

Length of comparison data. The length of the data to compare to the function information. The length of the comparison data must be valid for the function control key that is specified.

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

Function Control Keys

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

Key	Type	Field
1	CHAR(1)	Function category
3	CHAR(30)	Function product ID
4	CHAR(30)	Function group ID
5	CHAR(30)	Function ID

Field Descriptions

Function category. The category of the function.

The possible values are:

1 Locally managed client functions within iSeries Navigator are selected.
2 Locally managed client functions not within iSeries Navigator are selected.
3 Host functions are selected.
4 Centrally managed client functions within iSeries Navigator are selected.
5 Centrally managed client functions not within iSeries Navigator are selected.
7 All locally managed client functions are selected. This includes all functions in categories 1 and 2.
8 All centrally managed client functions are selected. This includes all functions in categories 4 and 5.
9 All client functions are selected. This includes all functions in categories 1, 2, 4, and 5.

Function group ID. All functions that have this function group are selected. The special value of *NONE can be specified to select functions that do not have a function group specified.

Function ID. The ID of the functions to be selected.

The following can be specified for the function ID:

*generic** All function IDs that begin with the generic string are selected.
function ID The specific function ID are selected.

Function product ID. All functions that have this function product are selected.

Error Messages

Message ID	Error Message Text
CPF2204 E	User profile &1 not found.
CPF229D E	Operation not allowed on user profile &1.
CPF3C19 E	Error occurred with receiver variable specified.
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.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C81 E	Value for key &1 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C90 E	Literal value cannot be changed.
CPF3CD9 E	Requested function cannot be performed at this time.

Message ID	Error Message Text
CPF3CDA E	Registration facility repository not available for use.
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.
CPF3CE7 E	Number of selection criteria entries 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.
CPF9810 E	Library &1 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Top | Security APIs | 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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI

DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for downloading and printing publications

Permissions for the use of the information you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce this information for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of this information, or any portion thereof, without the express consent of IBM(R).

Commercial Use: You may reproduce, distribute and display this information solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of this information, or reproduce, distribute or display this information or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the information or any data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the information is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THIS INFORMATION. THE INFORMATION IS PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

By downloading or printing information from this site, you have indicated your agreement with these terms and conditions.

Code disclaimer information

This document contains programming examples.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM(R), ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Printed in USA