IBM
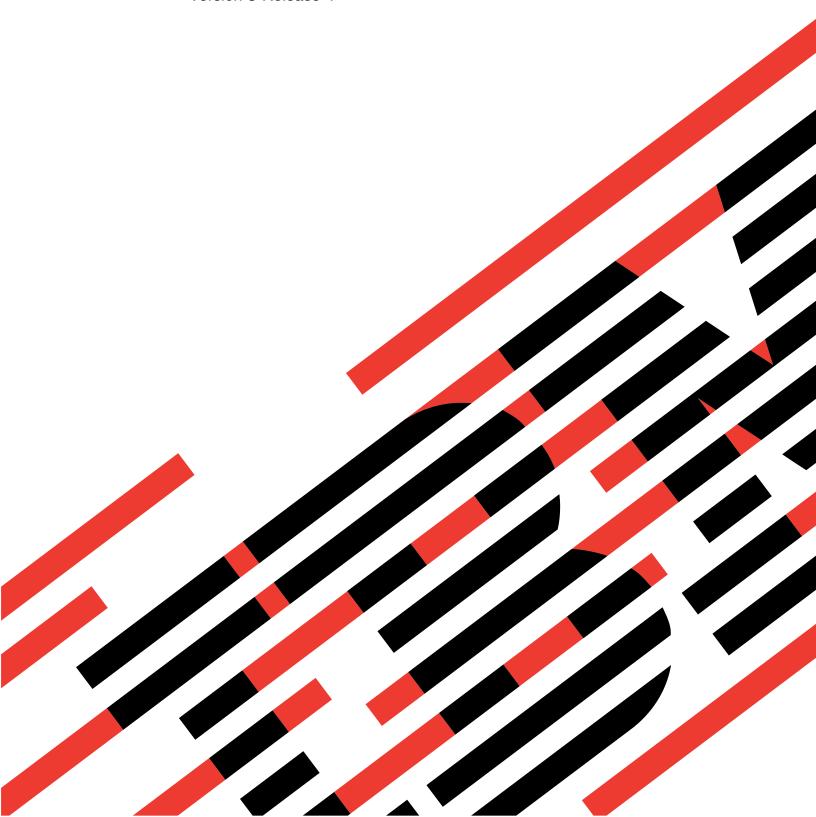
IBM Systems - iSeries

Security -- Security-related APIs

Version 5 Release 4

IBM

IBM Systems - iSeries
Security -- Security-related APIs
Version 5 Release 4

IBM

> **Note**
>
> Before using this information and the product it supports, be sure to read the information in
> "Notices," on page 195.

Sixth Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (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

Security-related APIs

The i5/OS$^{(TM)}$ security-related APIs allow you to:

- Perform many of the security functions through a program interface. You can use APIs instead of CL commands.
- Combine many individual jobs into a single server or overhead job without compromising system security.

These APIs can be used to consolidate server jobs to reduce processing time and storage use because the system performs job management tasks for only one job. They also speed response time for system users.

For general information about system security, see the Security topic.

The security-related APIs are:

- "Add Verifier (QYDOADDV, QydoAddVerifier) API" on page 3 (QYDOADDV, QydoAddVerifier)) adds a certificate to the local system's *SIGNATUREVERIFICATION certificate store that the local system can use later to verify the integrity of objects on the system.
- "Change Previous Sign-On Date (QSYCHGPR) API" on page 5 (QSYCHGPR) changes the previous sign-on date and time to the current date and time for the current user of the job.
- "Change Service Tools User ID (QSYCHGDS) API" on page 6 (QSYCHGDS) changes the ID name or the password (or both) for service tools user IDs.
- "Change User Password (QSYCHGPW) API" on page 9 (QSYCHGPW) changes a user's password.
- "Change User Profile UID or GID (QSYCHGID) API" on page 13 (QSYCHGID) changes the user ID (UID) or group ID (GID) value for a user profile object.
- "Check Encrypted User Password (QSYCUPWD) API" on page 14 (QSYCUPWD) checks to see if the encrypted password data for the specified user profile on the system on which this API is run is the same as the encrypted password data for the user on the system where the Retrieve Encrypted User Password (QSYRUPWD) API was run.
- "Check Profile Token User (QSYCHKTU, QsyChkPrfTknUser) API" on page 17 (QSYCHKTU, QsyChkPrfTknUser) verifies that the user profile associated with the token is the same as the current user profile in the thread.
- "Check System (QYDOCHKS, QydoCheckSystem) API" on page 18 (QYDOCHKS, QydoCheckSystem) checks key operating system object's signatures.
- "Check User Authority to an Object (QSYCUSRA) API" on page 21 (QSYCUSRA) returns an indication about a user's specified authority to an object.
- "Check User Special Authorities (QSYCUSRS) API" on page 23 (QSYCUSRS) returns an indication of a user's special authorities.
- "QwtClearJuid()—Clear Job User Identity" on page 25 (QwtClearJuid()) clears any job user identity that was previously set by the QwtSetJuid() function or by the Set Job User Identity (QWTSJUID) API.
- "Convert Authority Values to MI Value (QSYCVTA) API" on page 26 (QSYCVTA) converts authority values to the machine interface (MI) representation of the value.
- "Generate Profile Token (QSYGENPT) API" on page 30 (QSYGENPT) verifies that the caller has authority to generate a profile token for the requested profile and then generates a profile token.
- "Generate Profile Token (QsyGenPrfTkn) API" on page 28 (QsyGenPrfTkn) verifies that the caller has authority to generate a profile token for the requested profile and then generates a profile token.
- "Generate Profile Token Extended (QsyGenPrfTknE) API" on page 34 (QsyGenPrfTknE) verifies that the caller has authority to generate a profile token for the requested profile and then generates a profile token.

- "Generate Profile Token From Profile Token (QSYGENFT, QsyGenPrfTknFromPrfTkn) API" on page 37 (QSYGENFT, QsyGenPrfTknFromPrfTkn) generates a profile token using an existing profile token.
- "Get Profile Handle (QSYGETPH) API" on page 39 (QSYGETPH) validates a user ID and password, and creates an encrypted abbreviation called a profile handle for that user profile.
- "Get Profile Handle (QsyGetProfileHandle) API" on page 43 (QsyGetProfileHandle) validates user IDs and passwords and creates a profile handle, for use in jobs that run under more than one user profile.
- "Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API" on page 45 (QsyGetProfileHandleNoPwd) validates user IDs and creates a profile handle, for use in jobs that run under more than one user profile.
- "Get Profile Token Time Out (QSYGETPT, QsyGetPrfTknTimeOut) API" on page 48 (QSYGETPT, QsyGetPrfTknTimeOut) gets the number of seconds until a profile token is not valid.
- "Invalidate Profile Token (QSYINVPT, QsyInvalidatePrfTkn) API" on page 49 (QSYINVPT, QsyInvalidatePrfTkn) invalidates a profile token.
- "List Authorized Users (QSYLAUTU) API" on page 50 (QSYLAUTU) puts a list of authorized users of the system in a user space.
- "List Objects Secured by Authorization List (QSYLATLO) API" on page 57 (QSYLATLO) puts a list of objects secured by an authorization list in a user space.
- "List Objects That Adopt Owner Authority (QSYLOBJP) API" on page 64 (QSYLOBJP) puts a list of objects that adopt an owner's authority in a user space.
- "List Objects User Is Authorized to, Owns, or Is Primary Group of (QSYLOBJA) API" on page 69 (QSYLOBJA) puts a list of objects that a user is authorized to, owns, or is the primary group owner for into a user space.
- "List Users Authorized to Object (QSYLUSRA) API" on page 79 (QSYLUSRA) puts a list of users privately authorized to an object in a user space.
- "Open List of Authorized Users (QGYOLAUS) API" on page 84 (QGYOLAUS) provides information about the authorized users of the system.
- "Release Profile Handle (QSYRLSPH, QsyReleaseProfileHandle) API" on page 88 (QSYRLSPH, QsyReleaseProfileHandle) validates a given profile handle and then releases it.
- "Remove All Profile Tokens (QsyRemoveAllPrfTkns) API" on page 90 (QsyRemoveAllPrfTkns) provides an interface to remove all profiles on the system.
- "Remove All Profile Tokens For User (QsyRemoveAllPrfTknsForUser) API" on page 91 (QsyRemoveAllPrfTknsForUser) provides an interface to remove all profile tokens that have been generated for a specific user profile.
- "Remove Profile Token (QsyRemovePrfTkn) API" on page 92 (QsyRemovePrfTkn) removes the specified profile token.
- "Remove Profile Tokens (QSYRMVPT) API" on page 93 (QSYRMVPT) provides an interface to remove all profile tokens that have been generated for user profiles on the system, or to remove all profile tokens that have been generated for a specific user profile.
- "Reset Profile Attributes (QSYRESPA) API" on page 96 (QSYRESPA) resets four attributes of system-supplied user profiles.
- » Retrieve Authorization List Information (QSYRTVAI) returns the number of entries in use for the specified authorization list. «
- "Retrieve Authorized Users (QSYRAUTU) API" on page 97 (QSYRAUTU) returns a list of authorized user names on the system and information about those users.
- "Retrieve Encrypted User Password (QSYRUPWD) API" on page 102 (QSYRUPWD) returns to the caller the encrypted password for the specified user profile.
- "Retrieve Object Signatures (QYDORTVO, QydoRetrieveDigitalSignatures)API" on page 104 (QYDORTVO, QydoRetrieveDigitalSignatures) retrieves certificate information from a signed iSeries object.

- "Retrieve Objects Secured by Authorization List (QGYRATLO) API" on page 109 (QGYRATLO) provides a list of objects that are secured by an authorization list.
- "Retrieve Security Attributes (QSYRTVSA) API" on page 117 (QSYRTVSA) retrieves information about the current and pending security attributes of the system.
- "Retrieve User Authority to Object (QSYRUSRA) API" on page 123 (QSYRUSRA) returns the user's authority to an object.
- "Retrieve User Information (QSYRUSRI) API" on page 132 (QSYRUSRI) returns the information about a user.
- "Retrieve Users Authorized to an Object (QSYRTVUA) API" on page 147 (QSYRTVUA) provides information about the users who are authorized to an object.
- "Set Encrypted User Password (QSYSUPWD) API" on page 154 (QSYSUPWD) sets the encrypted password for the specified user profile by using the receiver variable that was retrieved by the Retrieve Encrypted User Password (QSYRUPWD) API.
- "Set Job User Identity (QWTSJUID) API" on page 156 (QWTSJUID) performs two operations that can be used to explicitly set the job user identity of the current job.
- "QwtSetJuid()—Set Job User Identity" on page 157 (QwtSetJuid()) sets the job user identity of the current job to the name of the current user profile of the job.
- "Set Profile Handle (QWTSETP, QsySetToProfileHandle) API" on page 158 (QWTSETP, QsySetToProfileHandle) switches the job to run under a new profile.
- "Set To Profile Token (QSYSETPT, QsySetToPrfTkn) API" on page 160 (QSYSETPT, QsySetToPrfTkn) validates the profile token and changes the current thread to run under the user and group profiles represented by the profile token.
- "Sign Buffer (QYDOSGNB, QydoSignBuffer)" on page 162 (QYDOSGNB, QydoSignBuffer) allows the local system to certify that the series of bytes being signed is trustworthy.
- "Sign Object (QYDOSGNO, QydoSignObject) API" on page 166 (QYDOSGNO, QydoSignObject) allows the local system to certify that the object being signed is trustworthy as of the time the object is being signed.
- "Verify Buffer (QYDOVFYB, QydoVerifyBuffer)" on page 174 (QYDOVFYB, QydoVerifyBuffer) allows the local system to verify that the series of bytes signed earlier has not been tampered with.
- "Verify Object (QYDOVFYO, QydoVerifyObject) API" on page 177 (QYDOVFYO, QydoVerifyObject) checks to see if an object has changed since it was signed.

APIs

These are the APIs for this category.

Add Verifier (QYDOADDV, QydoAddVerifier) API

Required Parameter Group:

1	Certificate path name	Input	Char(*)
2	Length of certificate path name	Input	Binary(4)
3	Format of certificate path name	Input	Char(8)
4	Certificate label	Input	Char(*)
5	Length of certificate label	Input	Binary(4)
6	Error code	I/O	Char(*)

Service Program Name: QYDOADD1
Default Public Authority: *USE
Threadsafe: No

The Add Verifier (OPM, QYDOADDV; ILE, QydoAddVerifier) API adds a certificate to the local system's *SIGNATUREVERIFICATION certificate store that the local system can use later to verify the integrity of objects on the system. This certificate represents the system or company that has signed objects that the local system will want to use. Object signatures are used to detect changes to an object that affect the integrity of that object. Object signatures also identify the origin of the object; that is, which system or company the object came from.

Note: If the certificate store does not exist, it will be created with a certificate store password of "VERIFYSIGNATURE". This password should be changed as soon as possible to a non-trivial password using the Digital Certificate Manager.

Authorities and Locks

Authority Required
> *ALLOBJ and *SECADM special authorities. Also the "allow certificate updates" must be set on the service tools menu.

Locks   Object containing certificate will be locked exclusive no read

Required Parameter Group

Certificate path name
> INPUT; CHAR(*)

> The path name of the stream file that has the certificate you wish to add to the *SIGNATUREVERIFICATION certificate store on the local system. This certificate store is a list of certificates the local system uses to verify the integrity of signed objects. If you are using format OBJN0100, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

Length of certificate path name
> INPUT; BINARY(4)

> The length of the contents of the certificate path name parameter. If the format of certificate path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of the certificate path name is OBJN0100, only the path name itself is included.

Format of certificate path name
> INPUT; CHAR(8)

> The format of the certificate path name parameter.

OBJN0100	The certificate path name is a simple path name.
OBJN0200	The certificate path name is an LG-type path name.

Certificate label
> INPUT; CHAR(*)

> Names the certificate that will be stored in the database. This label must be unique in the database; you cannot have another certificate with the same name in the database.

> This certificate should have been created by exporting a verification certificate from the *OBJECTSIGNING certificate store on the system that signed the objects or buffers to be verified. Exporting any other way will not be useable by this API. Digital Certificate Manager (DCM) can be used for several file formats including this format. DCM will need to be used if other file formats are used.

This certificate should not have been signed by a local Certificate Authority (CA). This API does not support adding CA certificates. DCM will need to be used to import CA certificates prior to using this API to add certificates from those CAs. The certificate stores come with several Internet CA certificates already installed.

Length of certificate label
> INPUT; BINARY(4)

> The length of the contents of the certificate label parameter.

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	User profile does not have *SECADM (or *ALLOBJ) special authority.
CPFA0A9 E	Object not found. Object is &1.
CPFB724 E	Option &2 of the operating system is required to work with object signatures.
CPFB73A E	The password for the certificate key database needs to be set.
CPF9EA2 E	Certificate is not in a supported format. This certificate may have been exported from the *SIGNATUREVERIFICATION certificate store instead of the *OBJECTSIGNING certificate store.
CPF9EA6 E	Function &1 cannot be used. The function specified is one that is currently prevented from being used.
CPF9EB0 E	Certificate with label &2 is already in the certificate store.
CPF9EB2 E	A Certificate Authority (CA) certificate cannot be added using this API.
CPF9EB3 E	The issuer of the certificate may not be in the certificate store. Certificate was not added.

API introduced: V5R2

Change Previous Sign-On Date (QSYCHGPR) API

Required Parameter:

1	Error Code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Change Previous Sign-On Date (QSYCHGPR) API changes the previous sign-on date and time to the current date and time for the user profile that is currently running. If a user has been swapped in using the Set Profile (QWTSETP) API, that user's previous sign-on date and time is the one that gets changed.

Authorities and Locks

None

Required Parameter

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
CPF2213 E	Not able to allocate user profile &1.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Change Service Tools User ID (QSYCHGDS) API

Required Parameter Group:

1	Requesting service tools user ID	Input	Char(*)
2	Requesting service tools user ID password	Input	Char(*)
3	Service tools user ID to be changed	Input	Char(10)
4	New service tools user ID	Input	Char(*)
5	New service tools user ID password	Input	Char(*)
6	Error code	I/O	Char(*)

Optional Parameter Group:

7	Length of requesting service tools user ID profile	Input	Bin(4)
8	Length of requesting service tools user ID password	Input	Bin(4)
9	CCSID of requesting service tools user ID password	Input	Bin(4)
10	Length of new service tools user ID	Input	Bin(4)
11	Length of new service tools user ID password	Input	Bin(4)
12	CCSID of new service tools user ID password	Input	Bin(4)

Default Public Authority: *EXCLUDE
Threadsafe: No

The Change Service tools user ID (QSYCHGDS) API changes the ID name or the password (or both) for service tools user IDs.

Calling this API from a command line may result in a security exposure since the plain text service tools user ID and password may appear in the job log. Therefore, you should avoid calling this API from the command line.

Authorities and Locks

When the requesting service tools user ID is different than the service tools user ID to be changed, the requesting service tools user ID must have the Service Tool user function privilege "Service Tool Security".

When the requesting service tools user ID is the same as the service tools user ID to be changed and the service tools user ID name is to be changed, the requesting service tools user ID must have the Service Tool user function privilege "Service Tool Security".

Required Parameter Group

Requesting service tools user ID
      INPUT; CHAR(*)

The requesting service tools user ID. This value is converted to uppercase. If the optional parameter group is not specified, a default of length 8 is used. The requesting service tools user ID parameter should be padded with trailing blank characters if necessary.

Requesting service tools user ID password
      INPUT; CHAR(*)

The password for the requesting service tools user ID.

Service tools user ID to be changed
      INPUT; CHAR(10)

The service tools user ID to be changed. This value is converted to uppercase. The service tools user ID to be changed parameter should be padded with trailing blank characters. You can use these special values for the service tools user ID to be changed:

SECURITY	Change the security capability profile.
FULL	Change the full capability profile.
BASIC	Change the basic capability profile.
SERVICE	Change the service capability profile.

New service tools user ID
      INPUT; CHAR(*)

The new service tools user ID or *SAME. This value is converted to uppercase. If the optional parameter group is not specified, a default length of 8 is used. The new service tools user ID parameter should be padded with trailing blank characters if necessary.

New service tools user ID password
      INPUT; CHAR(*)

The new service tools user ID password or *SAME.

If 128 character, case sensitive passwords are enabled for the service tools user IDs, then *SAME is not allowed.

Error code
      I/O; CHAR(*)

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

Optional Parameter Group

Length of requesting service tools user ID
      INPUT; BINARY(4)

The length, in bytes, of the ID contained in the requesting service tools ID parameter. If the optional parameter group is not specified, a default of 8 is used. The requesting service tools user ID parameter should be padded with trailing blank characters, if necessary, to the size specified by this parameter.

This parameter accepts values from 1 to 10.

Length of requesting service tools user ID password
      INPUT; BINARY(4)

The length, in bytes, of the password contained in the requesting service tools user ID password parameter. If the optional parameter group is not specified, a default of 8 is used. The requesting service tools user ID password parameter should be padded with trailing blank characters, if necessary, to the size specified by this parameter.

This parameter accepts values from 1 to 512. If 128 character, case sensitive passwords are not enabled for the service tools user IDs, the password is limited to 8 characters. If 128 character, case sensitive passwords are enabled for the service tools user IDs, then this password may be from 6 to 512 bytes. However, values greater than 128 should only be used if multi-byte characters are specified for the service tools user ID password. The number of characters, as interpreted by the CCSID of the service tools user ID password parameter, cannot exceed 128.

CCSID of requesting service tools user ID password
INPUT; BINARY(4)

The CCSID of the requesting service tools user ID password parameter. If the optional parameter group is not specified, CCSID 37 is used. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

The valid values are:

0	The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533	A valid CCSID in this range.

Length of new service tools user ID
INPUT; BINARY(4)

The length, in bytes, of the ID contained in the new service tools user ID parameter. If the optional parameter group is not specified, a default of 8 is used. The new service tools user ID parameter should be padded with trailing blank characters, if necessary, to the size specified by this parameter.

This parameter accepts values from 1 to 10.

Length of new service tools user ID password
INPUT; BINARY(4)

The length, in bytes, of the password contained in the new service tools user ID password parameter. If the optional parameter group is not specified, a default of 8 is used. The new service tools user ID password parameter should be padded with trailing blank characters, if necessary, to the size specified by this parameter.

This parameter accepts values from 6 to 512. If 128 character, case sensitive passwords are not enabled for the service tools user IDs, the password is limited to 8 characters. If 128 character, case sensitive passwords are enabled for the service tools user IDs, then this password may be from 6 to 512 bytes. However, values greater than 128 should only be used if multi-byte characters are specified for the service tools user ID password. The number of characters, as interpreted by the CCSID of the service tools user ID password parameter, cannot exceed 128.

CCSID of new service tools user ID password
INPUT; BINARY(4)

The CCSID of the new service tools user ID password parameter. If the optional parameter group is not specified, CCSID 37 is used. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

The valid values are:

0	The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533	A valid CCSID in this range.

Error Messages

Message ID	Error Message Text
CPF225B E	Service tools user ID to be changed is not correct.
CPF225C E	Requesting service tools user ID not correct.
CPF225D E	Requesting service tools user ID password not correct.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3BDE E	CCSID &1 not supported by API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF4AB3 E	Error changing service tools user ID. Reason code &1.
CPF4AB7 E	Service tools user ID password cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R1

Change User Password (QSYCHGPW) API

Required Parameter Group:

1	User ID	Input	Char(10)
2	Current password	Input	Char(*)
3	New password	Input	Char(*)
4	Error code	I/O	Char(*)

Optional Parameter Group:

5	Length of current password	Input	Bin(4)
6	CCSID of current password	Input	Bin(4)
7	Length of new password	Input	Bin(4)
8	CCSID of new password	Input	Bin(4)

Default Public Authority: *USE
Threadsafe: No

The Change User Password (QSYCHGPW) API changes a user's password. You must know the existing password that you want to change, unless you have *SECADM special authority and *OBJMGT and *USE authority to the user profile being changed.

This API provides support similar to the Change Password (CHGPWD) command.

Authorities and Locks

If the user ID parameter is not *CURRENT or the user ID of the user that is currently running, the caller of the API must have *SECADM special authority and *OBJMGT and *USE authorities to the user profile being changed to change the password. If the current password parameter is *NOPWD, the caller of the API must have *SECADM special authority and *OBJMGT and *USE authorities to the user profile being changed.

Required Parameter Group

User ID
    INPUT; CHAR(10)

The name of the user whose password is being changed.

You can specify the following special value:

*CURRENT   The password of the user currently running is changed.

Current password
    INPUT; CHAR(*)

The current password for the user. Verification is done to ensure this is the correct password for the user before the password is changed, unless *NOPWD is specified. All trailing blank and null characters are removed from the current password before it is verified.

You can specify the following special values:

*NONE   The user currently does not have a password or the password is not managed locally.
*NOPWD   The current password for the user is not verified before changing the password. The caller of the API must have *SECADM special authority and *OBJMGT and *USE authorities to the user profile being changed to specify this value.

New password
    INPUT; CHAR(*)

The new password for the user. Verification is done to ensure the new password meets the password composition rules of the system. All trailing blank and null characters are removed from the new password before it is verified.

You can specify the following special value:

*NONE   The user is changed to not have a password. This value is not allowed if *CURRENT, the user ID of the user that is currently running, or QSECOFR is specified on the user ID parameter.

If the local password management (LCLPWDMGT) value for the user profile specified on the user ID parameter is set to *NO, then the local i5/OS password will be set to *NONE. The password value specified in this parameter will be sent to other IBM products that do password synchronization (for example, iSeries Integration for Windows Server).

Error code
    I/O; CHAR(*)

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

Optional Parameter Group

Length of current password
    INPUT; BINARY(4)

The length, in bytes, of the password contained in the current password parameter. If the optional parameter group is not specified, a default of 10 is used. The current password parameter should be padded with trailing blank characters, if necessary, to the size specified by this parameter.

This parameter accepts values from 1 to 512; however, values greater than 128 should only be used if multi-byte characters are specified for the current password. The number of characters, as interpreted by the CCSID of the current password parameter, cannot exceed 128.

CCSID of current password
>
> INPUT; BINARY(4)
>
> The CCSID of the current password parameter. If the optional parameter group is not specified and the system is operating at password level 0 or 1, CCSID 37 is used. If the optional parameter group is not specified and the system is operating at password level 2 or 3, the default CCSID of the job is used to determine the CCSID of the data to be converted. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.
>
> The valid values are:

0	The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533	A valid CCSID in this range.
65535	When the system is operating at password level 0 or 1, CCSID 37 is used. When the system is operating at password level 2 or 3, this value is rejected.

Length of new password
>
> INPUT; BINARY(4)
>
> The length, in bytes, of the password contained in the new password parameter. If the optional parameter group is not specified, a default of 10 is used. The new password parameter should be padded with trailing blank characters, if necessary, to the size specified by this parameter.
>
> This parameter accepts values from 1 to 512; however, values greater than 128 should only be used if multi-byte characters are specified for the new password. The number of characters, as interpreted by the CCSID of the new password parameter, cannot exceed 128.

CCSID of new password
>
> INPUT; BINARY(4)
>
> The CCSID of the new password parameter. If the optional parameter group is not specified and the system is operating at password level 0 or 1, CCSID 37 is used. If the optional parameter group is not specified and the system is operating at password level 2 or 3, the default CCSID of the job is used to determine the CCSID of the data to be converted. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.
>
> The valid values are:

0	The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533	A valid CCSID in this range.
65535	When the system is operating at password level 0 or 1, CCSID 37 is used. When the system is operating at password level 2 or 3, this value is rejected.

Usage Notes

If the caller of the API:

- Enters the wrong password for the user, and
- Exceeds the maximum number of times allowed by the system value QMAXSIGN, and
- The system value QMAXSGNACN is set to disable user profiles,

then the user profile specified on the user parameter is disabled.

You cannot specify the following user ID profile names for the user parameter:

```
QAUTPROF   QCLUMGT    QCLUSTER   QCOLSRV
QDBSHR     QDBSHRDO   QDIRSRV    QDFTOWN
QDLFM      QDOC       QDSNX      QFNC
QGATE      QIPP       QLPAUTO    QLPINSTALL
```

```
QMGTC      QMSF       QNFSANON   QNETSPLF
QNTP       QPEX       QPM400     QSNADS
QSPL       QSPLJOB    QSRVAGT    QSYS
QTCM       QTCP       QTFTP      QTMHHTP1
QTSTRQS    QYCMCIMOM  QYPSJSVR
```

When the new password is checked to ensure it meets the password composition rules for the system, only one error is returned per API call. Therefore, if the new password fails more than one of the rules, multiple calls to the API are needed to determine a correct new password.

If *NOPWD is specified for the current password, then the QPWDPOSDIF (Limit password character positions) system value cannot be checked. This system value determines whether the characters in the same position in the current and new password must be different. This value cannot be checked without the current password value.

You should avoid calling this API from a command line. If this API is called from CL and CL commands are being logged for the job or CL program, the call parameters for the API are logged in the job log. This means the passwords appear in the job log.

If the optional parameter group is not specified, the current and new password lengths default to 10 and the CCSID of the current and new passwords default to 37. These are the values that were used by the QSYCHGPW API prior to the addition of the optional parameter group.

You cannot specify a password length greater than 10 unless the system is operating at a password level of 2 or 3.

Error Messages

Message ID	Error Message Text
CPD2201 E	System user profile cannot be changed.
CPD2356 E	New password cannot be same as current password.
CPF0001 E	Error found on &1 command.
CPF22C0 E	Password does not meet password rules. Return code &1.
CPF22C2 E	Password less than &1 characters.
CPF22C3 E	Password longer than &1 characters.
CPF22C4 E	Password matches one of 32 previous passwords.
CPF22C5 E	Password contains one of the following: &1.
CPF22C6 E	Password contains two numbers next to each other.
CPF22C7 E	Password contains a character used more than once.
CPF22C8 E	Same character in same position as previous password.
CPF22C9 E	Password must contain a number.
CPF22D0 E	Password contains a character repeated consecutively.
CPF22D1 E	Password cannot be same as user ID.
CPF22D2 E	Password approval program &1 not found.
CPF22D3 E	Password approval program signaled an error.
CPF22D4 E	Not allowed to use password approval program.
CPF22D5 E	Parameters in password approval program not correct.
CPF22E2 E	Password not correct for user profile &1.
CPF22E3 E	User profile &1 is disabled.
CPF22F5 E	Value &1 for new password not allowed.
CPF22F6 E	New password cannot be *NONE.
CPF2203 E	User profile &1 not correct.
CPF2213 E	Not able to allocate user profile &1.
CPF222E E	&1 special authority is required.
CPF2225 E	Not able to allocate internal system object.
CPF2292 E	*SECADM required to create or change user profiles.

Message ID	Error Message Text
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Change User Profile UID or GID (QSYCHGID) API

Required Parameter Group:

1	User profile name	Input	Char(10)
2	User ID number (UID)	Input	Binary(4)
3	Group ID number (GID)	Input	Binary(4)
4	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Change User Profile UID or GID (QSYCHGID) API provides an interface to change the user ID number (UID) or group ID number (GID) for a user profile. The UID or GID value for any user profile on the system may be changed. If the UID value is changed and the user profile owns objects in a directory (not including objects in the QSYS.LIB or QDLS file system), then the UID information for these objects is also changed. If the GID value is changed and the user profile is the primary group for objects in a directory, then the GID information for these objects is also changed. The UID or GID of a profile that is active in a process can be changed only when the system is in restricted state. (For example, the system would probably have to be in restricted state to change the UID for the QSYS user profile.) However, the UID of the user profile currently running cannot be changed, and the GID of the groups for the user profile currently running cannot be changed.

Authorities and Locks

User Profile Authority
　　*ALLOBJ and *SECADM

User Profile Lock
　　*EXCL

Required Parameter Group

User profile name
　　INPUT; CHAR(10)

　　The name of the user profile whose UID or GID is to be changed.

User ID number (UID)
　　INPUT; BINARY(4)

The new UID for the user profile.

This field must contain one of the following values:

1 to 4294967294	The new UID value.
4294967295	The UID for this user profile does not change. This value is the same as X'FFFFFFFF' or -1 in languages that do not support unsigned integers.

Group ID number (GID)
INPUT; BINARY(4)

The new GID for the user profile.

This field must contain one of the following values:

1 to 4294967294	The new GID.
4294967295	The GID for this user profile does not change. This value is the same as X'FFFFFFFF' or -1 in languages that do not support unsigned integers.
0	This user profile will no longer have a GID.

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
CPFA1C8 D	Error occurred while attempting to change the UID or GID information.
CPF22CE E	The &1 value &2 is used by another user profile.
CPF22DB E	The user profile being changed must have a GID.
CPF22DE E	Not allowed to change the UID or GID of user profile &1.
CPF2203 E	User profile &1 not correct.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF222E E	&1 special authority is required.
CPF2225 E	Not able to allocate internal system object.
CPF224B E	&1 value is not valid.
CPF224C E	Cannot change the UID value for QSECOFR.
CPF224D E	User profile &1 cannot have a GID.
CPF3CF1 E	Error code parameter not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Check Encrypted User Password (QSYCUPWD) API

Required Parameter Group:

1	Encrypted password return code	Output	Char(1)
2	Receiver variable from QSYRUPWD	Input	Char(*)
3	Format	Input	Char(8)
4	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: No

The Check Encrypted User Password (QSYCUPWD) API checks to see if the encrypted password data for the specified user profile on the system on which this API is run is the same as the encrypted password data for the user on the system where the Retrieve Encrypted User Password (QSYRUPWD) API was run.

The API does not check the iSeries Support for Windows Network Neighborhood (iSeries NetServer) encrypted password information. Only the encrypted passwords used to sign on from a sign-on display are checked.

The QSYCUPWD API follows this process:
- Verifies that the user calling this API is authorized.
- Verifies that the user profile specified in the receiver variable from QSYRUPWD parameter exists and is correct.
- If the user profile is disabled, the incorrect password count is incremented and the appropriate value is set in the encrypted password return code.
- If the password for the user profile is *NONE or expired, the appropriate value is set in the encrypted password return code. If the local password management (LCLPWDMGT) value for the user profile is *NO, then the password for the user profile will be *NONE.
- Checks to see if the encrypted passwords can be compared. If the passwords cannot be compared, the appropriate value is set in the encrypted password return code.

  The release versions and password levels must be compatible between the system on which this API is run and the system where the QSYRUPWD API was run to be able to compare the passwords. The passwords can be compared only if the user profile has a password for password level 0 or 1 on both systems or a password for password level 2 or 3 on both systems. If a system is at a release previous to V5R1M0, then the password for the user profile on that system is a password for password level 0 or 1.

  To determine if the user profile has a password for password level 0 or 1 or for password level 2 or 3, run either the Display Authorized Users (DSPAUTUSR) command and use the F11 key to see password level information, the Print User Profile (PRTUSRPRF) command using TYPE(*PWDLVL), or the Display User Profile (DSPUSRPRF) command using TYPE(*BASIC) to an outfile. These commands must be run on a V5R1M0 (or later) system.
- Compares the passwords. If the passwords do not match, the incorrect password count is incremented. The QMAXSIGN system value contains the maximum number of incorrect attempts to sign on. If the QMAXSGNACN system value is set to disable the user profile, repeated attempts to check the encrypted password when there is a mismatch will disable the user profile.

Authorities and Locks

User Profile Authority
        Caller of this API must have *ALLOBJ and *SECADM special authorities

API Public Authority
        *EXCLUDE

Required Parameter Group

Encrypted password return code
        OUTPUT; CHAR(1)

        Whether the encrypted password for the user profile on the system on which this API is run matches the encrypted password for the same user profile that is specified in the receiver variable from QSYRUPWD parameter. This parameter contains one of the following:

0	The passwords match.
1	The user profile on the system on which this API is run is disabled.
2	The password for the user on the system on which this API is run is *NONE.
3	The password for the user profile on the system on which this API is run is expired.
4	The passwords could not be compared.
9	The passwords do not match.

Receiver variable from QSYRUPWD
> INPUT; CHAR(*)

> The variable that is used to check the encrypted password for the user. The receiver variable from the QSYRUPWD API must be used as input to this API. For this API to successfully check the encrypted password for the user, the bytes returned value must be equal to the bytes available value in the input data. The input data must be retrieved from the receiver variable used by the QSYRUPWD API and cannot be changed in any way.

Format
> INPUT; CHAR(8)

> The name of the format that is used to check the user's encrypted password data. The following value is allowed:

"UPWD0100 Format"   Encrypted password will be checked.

Error code
> I/O; CHAR(*)

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

UPWD0100 Format

The following table describes the input variable that is to be passed as the second parameter to QSYCUPWD. This input variable must be the same data as the receiver variable that is returned by the QSYRUPWD API. The receiver variable, returned by the QSYRUPWD API, cannot be changed in any way prior to passing the data as input to the QSYCUPWD API. If this data is changed, the QSYCUPWD API will not be able to successfully check the password for the user. For detailed descriptions of the fields in the tables, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User profile name
18	12	CHAR(*)	Encrypted user password data

Field Descriptions

Bytes available. The number of bytes of data available when retrieved by the QSYRUPWD API. For the QSYCUPWD API to successfully check the encrypted password for the user, this value must be equal to the bytes returned value. If the bytes available field is greater than the bytes returned field, this input cannot be used to successfully check the encrypted password for the user.

Bytes returned. The number of bytes of data.

Encrypted user password data. The encrypted password data for the user profile.

User profile name. The name of the user profile for which the password will be checked.

Error Messages

Message ID	Error Message Text
CPF2203 E	User profile &1 not correct.
CPF2225 E	Not able to allocate internal system object.
CPF222E E	&1 special authority is required.
CPF3C21 E	Format name &1 is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF4AB2 E	Receiver variable from QSYRUPWD has been altered.
CPF9801 E	Object &2 in library &3 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

Check Profile Token User (QSYCHKTU, QsyChkPrfTknUser) API

》

Required Parameter Group for QSYCHKTU:

1	Result	Output	Bin(4)
2	Profile token	Input	Char(32)
3	Error Code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

Syntax for QsyChkPrfTknUser:

```
#include <qsyptkn.h>

void QsyChkPrfTknUser
     (int           *Result,
      unsigned char *Profile_token,
      void          *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

《

The Check Profile Token User (OPM, QSYCHKTU; ILE, QsyChkPrfTknUser) API verifies that the user profile associated with the token is the same as the current user profile in the thread. No other attributes associated with the token are compared with the attributes of the current thread.

Authorities and Locks

None

Required Parameter Group

Result OUTPUT; BIN(4)

The results from the check. If 1 is returned, the profile associated with the token is the same as the current user profile in the thread. If 0 is returned, the profile associated with the token is different from the current user profile in the thread.

Profile token
> INPUT; CHAR(32)

> The profile token to be checked.

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
CPF2225 E	Not able to allocate internal system object.
CPF2274 E	Profile token is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &, entered for this API was not valid.
CPF9872 E	Program or service program & in library &2 ended. Reason code &3.

API introduced: V5R1

Check System (QYDOCHKS, QydoCheckSystem) API

Optional Parameter Group:

1	Results path name	Input	Char(*)
2	Length of results path name	Input	Binary(4)
3	Format of results path name	Input	Char(8)
4	Format of content of results file	Input	Char (8)
5	Error code	I/O	Char(*)

Service Program Name: QYDOCHK1
Default Public Authority: *USE
Threadsafe: No

The Check System (OPM, QYDOCHKS; ILE, QydoCheckSystem) API checks to see if any key operating system object has changed since it was signed. If any of these objects is unsigned, it is reported as an error. Only signatures from a system trusted source are valid.
Note: This API can take several hours to complete.

Authorities and Locks

Authority Required
> *AUDIT special authority is required.

> See open() API for the authority needed to the results path name. The file is open for append and is created if it does not already exist.

Locks   Object will be locked shared allow read.

Optional Parameter Group

Results path name.
> INPUT; CHAR(*)

> The path name of the object you want to contain the results on this call. This object may not be in a library (that is, may not be under the /QSYS.LIB directory). The name may be relative to the current directory or may specify the entire path name. For example, to store results in a file called SIGNED.LST in the MYDIR directory, the results path name would be '/MYDIR/SIGNED.LST'. If you are using format OBJN0100, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

> If this is an existing file, results will be appended to the end of the file. Otherwise, a new file will be created.

> The default is not to have a results file.

Length of results path name.
> INPUT; BINARY(4)

> The length of the results path name. A 0 (zero) length means no results files are used, and the results path name and format of results path name parameter values are not used. If the format of the results path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of the results path name is OBJN0100, only the path name itself is included.

Format of results path name
> INPUT; CHAR(8)

> Format of the results path name

OBJN0100	The results path name is a simple path name.
OBJN0200	The results path name is an LG-type path name.

Format of content of results path name
> INPUT; CHAR(8)

> The format of the contents of the file containing the results of this call.

RSLT0100	The basic information is returned for each object specified by the object path name parameter.

Error code
> I/O; CHAR(*)

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

RSLT0100 format

The following table describes the order and format of the data returned in the RSLT0100 format. This data is repeated for each object that was attempted to be verified. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 20.

Note: All data in this file will be in CCSID 1200. New files will be created in this CCSID. If an existing file is named that has a different CCSID, an error will be reported.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(7)	Message identifier

Offset		Type	Field
Dec	Hex		
7	7	CHAR(9)	Reserved
16	10	CHAR(8)	Date
24	18	CHAR(8)	Reserved
32	20	CHAR(1)	Operation type
33	21	CHAR(15)	Operation type description
48	30	CHAR(8)	Reserved
56	38	CHAR(*)	Fully qualified object name

Field Descriptions

Date. The date the operation took place. The format will be YYYYMMDD. For example, June 30, 2002 will be 20020630.

Fully qualified object name. The simple path name from the root to the object whose signature is being verified. The field will be terminated with a new line character.

Message identifier. The error message used to report failure. This field is blank if no error was detected for this object.

Operation type. The operation that was attempted.

0   Signing operation
1   Verifying operation
2   Checking operation

Operation type description. Short word description of the operation that was attempted.

Reserved. This field currently is not used. It is filled with blanks.

Usage Notes

The following messages can be sent to the joblog and added to records in the Results Path Name.

Message ID	Message Text
CPFB722 D	Object not signed.
CPFB723 D	Object signed, but signature is not valid.
CPFB72A D	The object had no trusted signatures on the object.
CPFB72B D	Object not found.
CPFB72C D	The object cannot currently be signed or verified.

Error Messages

Message ID	Error Message Text
CPF222E E	&1 special authority is required.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9EA6 E	QVFYOBJRST system value does not verify object signatures during restore at its current setting.
CPFA08D E	Request information value is not valid.
CPFA0A4 E	Too many open files for process.
CPFA0AA E	Error occurred while attempting to obtain space.

Message ID	Error Message Text
CPFA0D4 E	File system error occurred.
CPFB735 E	The digital signature API parameter &1 is not large enough.
CPFB736 E	The digital signature API parameter &1 is not small enough.
CPFB737 E	The digital signature API parameter &1 is a null pointer.
CPFB738 E	The digital signature API parameter &1 is not a valid format type.
CPFB739 E	The digital signature API parameter &1 is out of range.
CPFB740 E	The format name for the pathname is not valid.
CPFB741 E	The length of the path name parameter is not valid.
CPFB744 E	The format of the results file for the digital signing API is an incorrect value.
CPFB745 E	The format name for the results file path name is not valid.
CPFB746 E	The results file path name length is not large enough.
CPFB749 E	Object signature operation ended abnormally. &1 objects attempted, &2 objects successfully processed.
CPFB74D E	Results file could not be used.

API introduced: V5R3

Check User Authority to an Object (QSYCUSRA) API

Required Parameter Group:

1	Authority indicator	Output	Char(1)
2	User profile name	Input	Char(10)
3	Qualified object name	Input	Char(20)
4	Object type	Input	Char(10)
5	Authority	Input	Char(*)
6	Number of authorities	Input	Binary(4)
7	Call level	Input	Binary(4)
8	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Check User Authority to Object (QSYCUSRA) API provides an indication of whether the user has the specified authority to an object.

Authorities and Locks

The following authority is required for the user calling this API, unless the user profile name parameter is *CURRENT or the name of the profile that is currently running, the caller owns the object, or the object is an authorization list:

- *OBJMGT authority to the object.
- *READ authority to the user profile.

If the user profile is *CURRENT or the name of the profile that is running currently, the authority to the user includes any authority specified on the object (private, group, authorization list, or public) plus any program adopted authority. If the user profile is not *CURRENT or the name of the profile that is running currently, the authority available to the user is the authority specified on the object.

Adopted authority is authority given to the user by the program for the duration of that program. If previous programs in the program stack adopt their owner's authority, the adopted authority for the current program is the accumulated adopted authority from all other programs in the program stack that adopt authority.

Required Parameter Group

Authority indicator
> OUTPUT; CHAR(1)

> Whether the user has the specified authority to the object. The field contains one of the following:

Y   The user has the specified authority.
N   The user does not have the specified authority.

User profile name
> INPUT; CHAR(10)

> The name of the user whose authority is checked.

> You can specify the following special value:

CURRENT   Checks the authority of the current user to the specified object.

Qualified object name
> INPUT; CHAR(20)

> The name of the object whose authority is checked. The first 10 characters specify the object name; the second 10 characters specify the library. You can use these special values for the library name:

CURLIB   The current library is used to locate the object. If there is no current library, QGPL (general purpose library) is used.
LIBL   The library list is used to locate the object.

Object type
> INPUT; CHAR(10)

> The type of object whose authority is checked.

Authority
> INPUT; CHAR(*)

> The authority to check for. This parameter can contain up to eleven 10-character fields. The following identifies the type of authority the user has to the object:

EXCLUDE   Exclude authority. If this value is specified, no other values can be specified.
ALL   All authority.
CHANGE   Change authority.
USE   Use authority.
AUTLMGT   Authorization list management authority. This value is only valid if the object type is *AUTL.
OBJALTER   Object alter authority.
OBJOPR   Object operational authority.
OBJMGT   Object management authority.
OBJEXIST   Object existence authority.
OBJREF   Object reference authority.
READ   Read authority.
ADD   Add authority.
UPD   Update authority.
DLT   Delete authority.
*EXECUTE   Execute authority.

Number of authorities
> INPUT; BINARY(4)

The number of authorities specified in the authority parameter. You can specify 1 through 11 authorities.

Call level

INPUT; BINARY(4)

The number of call levels to back up in the program stack to do the authority check. For example, if the program that calls this API adopts authority, you would probably not want the authority check to use the adopted authority. Therefore, the authority check should be done at the call level previous to the current level. This parameter should then contain a 1. You can check the authority at the various call levels by signifying a numeric equivalent to the call level. For example, to check the authority at the current call level, specify a 0; to check the authority at the previous call level, specify a 1.

This parameter is only used if the user profile name parameter is *CURRENT or the current user for the job.

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
CPF22FA E	Authority value &1 not valid.
CPF22FB E	Must specify *EXCLUDE or *AUTL as only authority value.
CPF22F7 E	Number of authorities must be between 1 and &1.
CPF22F9 E	Call level &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3C31 E	Object type &1 is not valid.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9814 E	Device &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Check User Special Authorities (QSYCUSRS) API

Required Parameter Group:

1	Authority indicator	Output	Char(1)
2	User profile name	Input	Char(10)
3	Special authority	Input	Char(*)

4	Number of authorities	Input	Binary(4)
5	Call level	Input	Binary(4)
6	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Check User Special Authorities (QSYCUSRS) API provides an indication of whether the user has the specified special authorities.

Authorities and Locks

User Profile Authority
　　　*READ

When the API checks for special authorities and the user profile name parameter is *CURRENT or the user who is currently running, the special authorities available to the user include any special authorities the user or the group has, and any program adopted special authorities. If the user profile specified is not the user currently running, then the special authorities available to the user are only the special authorities the user and his group have.

If previous programs in the program stack adopt their owner's authority, the adopted authority for the current program is the accumulated adopted authority from all other programs in the program stack that adopt authority.

Required Parameter Group

Authority indicator
　　　OUTPUT; CHAR(1)

　　　Whether the user has the specified special authorities.

　　　This parameter contains one of the following:

Y 　The user has the specified special authorities.
N 　The user does not have the specified special authorities.

User profile name
　　　INPUT; CHAR(10)

　　　The name of the user whose special authorities are checked.

　　　You can specify the following special value:

CURRENT 　　　The special authorities for the user currently running are checked.

Special authority
　　　INPUT; CHAR(*)

　　　The special authorities checked for the user. This parameter can contain up to eight 10-character fields.
　　　Each of the 10-character fields can contain one of the following special values.

ALLOBJ 　　　All object special authority.
AUDIT 　　　Audit special authority.
IOSYSCFG 　　　Input/output system configuration special authority.
JOBCTL 　　　Job control special authority.
SAVSYS 　　　Save system special authority.
SECADM 　　　Security administrator special authority.

SERVICE	Service special authority.
SPLCTL	Spool control special authority.

Number of authorities
> INPUT; BINARY(4)

> The number of special authorities specified in the special authority parameter. You can specify 1 through 8.

Call level
> INPUT; BINARY(4)

> The number of call levels to back up in the program stack to do the authority check. For example, if the program that calls this API adopts authority, you would probably not want the authority check to use the adopted authority. Therefore, the authority check should be done at the call level previous to the current level. This parameter should then contain a 1. You can check the authority at the various call levels by signifying a numeric equivalent to the call level. For example, to check the authority at the current call level, specify a 0; to check the authority at the previous call level, specify a 1.

> This parameter is only used if the user profile name parameter is *CURRENT, or the current user name for the job.

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
CPF22F7 E	Number of authorities must be between 1 and &1.
CPF22F8 E	Special authority value &1 not valid.
CPF22F9 E	Call level &1 not valid.
CPF2203 E	User profile &1 not correct.
CPF2225 E	Not able to allocate internal system object.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

QwtClearJuid()—Clear Job User Identity

Syntax

```
#include <qwtjuid.h>

int QwtClearJuid(void);
```

Service Program Name: QWTJUID
Default Public Authority: *EXCLUDE
Threadsafe: No; see "Usage Notes."

The**QwtClearJuid()** function clears any job user identity that was previously set by the QwtSetJuid()
function or by the Set Job User Identity (QWTSJUID) API. This function may only be called when the job
is running single threaded. The job user identity then defaults to the user profile that the job is currently
running under.

Parameters

None

Authorities and Locks

If the job user identity is currently set, then either *USE authority to the user profile associated with the
job user identity or all object (*ALLOBJ) special authority is required. If the job user identity is not
already set, then no authorization is required.

Return Value

[EPERM]

>Operation not permitted.

>You must have appropriate privileges or be the owner of the object or other resource to do the
requested operation.

>Function not allowed.

>Function not allowed while running multithreaded.

Usage Notes

The **QwtClearJuid()** function may be called in a job that allows multiple threads, but only while it is
running single threaded. It explicitly denies access if any secondary threads are active.

API introduced: V4R3

Convert Authority Values to MI Value (QSYCVTA) API

Required Parameter Group:

1	Converted authority value	Output	Char(2)
2	Authority special value	Input	Char(*)
3	Number of authorities	Input	Binary(4)
4	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Convert Authority Values to MI Value (QSYCVTA) API converts the special values indicating
authority to the corresponding machine interface (MI) representation of that value.

Required Parameter Group

Converted authority value
>   OUTPUT; CHAR(2)

>   The MI representation of the authority special value in hexadecimal format.

Authority special value
>   INPUT; CHAR(*)

>   The authority special values that are converted. The converted value is the cumulative value of all authority special values specified. This parameter can contain up to eleven 10-character fields. Each of the 10-character fields can contain one of the following special values. The following identifies the authority special values that are converted to the corresponding MI representation of that value.

*ADD	Add authority.
*ALL	All authority.
*AUTL	Authorization list authority. If this value is specified, no other values can specified. This authority value is only valid for *PUBLIC authority on an object secured by an authorization list.
*AUTLMGT	Authorization list management authority.
*CHANGE	Change authority.
*DLT	Delete authority.
*EXECUTE	Execute authority.
*EXCLUDE	Exclude authority. If this value is specified, no other values can be specified.
*OBJALTER	Object alter authority.
*OBJEXIST	Object existence authority.
*OBJMGT	Object management authority.
*OBJOPR	Object operational authority.
*OBJREF	Object reference authority.
*READ	Read authority.
*UPD	Update authority.
*USE	Use authority.

Number of authorities
>   INPUT; BINARY(4)

>   The number of authority special values specified in the authority special value parameter. You can specify 1 through 11.

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
CPF22FA E	Authority value &1 not valid.
CPF22FB E	Must specify *EXCLUDE or *AUTL as only authority value.
CPF22F7 E	Number of authorities must be between 1 and &1.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Generate Profile Token (QsyGenPrfTkn) API

≫

Syntax for QsyGenPrfTkn:

```
#include <qsyptkn.h>

void QsyGenPrfTkn
      (unsigned char  *Profile_token,
       char           *User_profile_name,
       char           *User_password,
       int            *Time_out_interval,
       char           *Profile_token_type,
       void           *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Generate Profile Token (QsyGenPrfTkn) API verifies that the caller has authority to generate a profile token for the requested profile and then generates a profile token. This profile token can be passed to one or more additional processes which can then use it to perform tasks on behalf of the authenticated user.

This API requires a special value to be specified for the user password parameter. If you need to validate a user password, see the Generate Profile Token Extended (QsyGenPrfTknE) API.

The Generate Profile Token API follows this process:
- Verifies that the user ID and password value are correct. Incorrect password values and special cases are handled as follows:

  - *NOPWD is not allowed if the user profile name is the name of the user profile running currently.
  - To obtain a profile token for a profile that does not have a password, specify *NOPWD, *NOPWDCHK or *NOPWDSTS for the password parameter.
    You cannot obtain a profile token for the following system-supplied user profiles:

QAUTPROF	QDLFM	QMSF	QSNADS	QTSTRQS
QCLUMGT	QDOC	QNETSPLF	QSPL	
QCOLSRV	QDSNX	QNFSANON	QSPLJOB	
QDBSHR	QFNC	QNTP	QSRVAGT	
QDBSHRDO	QGATE	QPEX	QSYS	
QDFTOWN	QLPAUTO	QPM400	QTCP	
QDIRSRV	QLPINSTALL	QRJE	QTFTP	

  - To obtain a profile token for a profile that is disabled, specify *NOPWDCHK for the password parameter.
  - To obtain a profile token when the password is expired, specify *NOPWDCHK or *NOPWDSTS for the password parameter.
- Generates the profile token designating the user's authorities.

  The maximum number of profile tokens that can be generated is approximately 2,000,000 per system; after that, the space to store them is full. Message CPF4AAA is sent to the application, and no more profile tokens can be generated until one is removed.
- Updates the last-used date for the user and its group profiles.
- Resets the signon attempts not valid count to zero when a profile token is successfully generated for a user.

- If security-related events are being audited, adds an entry to the QAUDJRN audit journal to indicate that a profile token is created.

Authorities and Locks

API Public Authority
> *USE

User profile authority
> *USE

User Profile Lock
> *LSRD

Required Parameter Group

Profile token
> OUTPUT; CHAR(32)
>
> The profile token that is generated.

User profile name
> INPUT; CHAR(10)
>
> The name of the user for which to generate the profile token.

User password
> INPUT; CHAR(10)
>
> The password value used to generate the profile token.
>
> Only special values are allowed for this parameter. A special value must be a 10 character, blank padded value in CCSID 37.
>
> One of the following special values must be specified:

*NOPWD   The user requesting the profile token must have *USE authority to the user profile.

A profile token does not get created for a disabled user profile.

A profile token does not get created for a user profile with an expired password.

This value is not allowed if the name of the currently running profile is specified for the user profile name parameter.

*NOPWDCHK   The user requesting the profile token must have *USE authority to the user profile.

If the profile is disabled, the user requesting the profile token must have *ALLOBJ and *SECADM special authorities to get a token.

If the password is expired, the user requesting the profile token must have *ALLOBJ and *SECADM special authorities to get a token.

*NOPWDSTS   The user requesting the profile token must have *USE authority to the user profile.

A profile token does not get created for a disabled user profile.

If the password is expired, the user requesting the profile token must have *ALLOBJ and *SECADM special authorities to get a token.

Time out interval
> INPUT; BINARY(4)
>
> The time before the profile token times out.

You can specify one of the following values:

-1	Use system default value (3600 seconds)
1-3600	Time out value in seconds.

Profile token type
> INPUT; CHAR(1)

> The type of the profile token to be generated.

> You can specify one of the following values:

1	Single-use profile token. A single-use profile token can be used only on the Set To Profile Token (QSYSETPT; QsySetToProfileToken) API once and cannot be used to generate new profile tokens.
2	Multiple-use profile token. A multiple-use profile token can be used on the Set To Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times, but cannot be used to generate new profile tokens.
3	Multiple-use, regenerable profile token. A multiple-use, regenerable profile token can be used on the Set To Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times and can be used to generate a new single-use, multiple-use, or multiple-use, regenerable profile token.

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
CPF22E3 E	User profile &1 is disabled.
CPF22E4 E	Password for user profile &1 has expired.
CPF22E9 E	*USE authority to user profile &1 required.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF227F E	*NOPWD not allowed for current user.
CPF3CF1 E	Error code parameter not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF4AAA E	Maximum number of profile tokens have been generated.
CPF4AAB E	Time out value not valid.
CPF4AAD E	Profile token type not valid.
CPF4AB8 E	Insufficient authority for user profile &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R5

Top | Security APIs | APIs by category

Generate Profile Token (QSYGENPT) API

Required Parameter Group:

1	Profile token	Output	Char(32)

30 IBM Systems - iSeries: Security -- Security-related APIs

2	User profile name	Input	Char(10)
3	User password	Input	Char(*)
4	Time out interval	Input	Bin(4)
5	Profile token type	Input	Char(1)
6	Error code	I/O	Char(*)

Optional Parameter Group:

7	Length of user password	Input	Bin(4)
8	CCSID of user password	Input	Bin(4)

Default Public Authority: *USE
Service Program: QSYPTKN
Threadsafe: Yes

The Generate Profile Token (OPM, QSYGENPT) API verifies that the caller has authority to generate a profile token for the requested profile and then generates a profile token. This profile token can be passed to one or more additional processes which can then use it to perform tasks on behalf of the authenticated user.

The Generate Profile Token API follows this process:

- Verifies that the user ID and password value are correct. Incorrect password values and special cases are handled as follows:

  - If the password is not correct, the incorrect password count is increased. (The QMAXSIGN system value contains the maximum number of incorrect attempts to sign on.) If the QMAXSGNACN system value is set to disable the user profile, repeated attempts to generate a profile token using an incorrect password disables the user ID. This keeps applications from methodically determining user passwords.
  - *NOPWD is not allowed if the user profile name is the name of the user profile running currently.
  - To obtain a profile token for a profile that does not have a password, specify *NOPWDCHK or *NOPWDSTS for the password parameter.
    You cannot obtain a profile token for the following system-supplied user profiles:

    ```
    QAUTPROF  QDLFM       QMSF      QSNADS   QTSTRQS
    QCLUMGT   QDOC        QNETSPLF  QSPL
    QCOLSRV   QDSNX       QNFSANON  QSPLJOB
    QDBSHR    QFNC        QNTP      QSRVAGT
    QDBSHRDO  QGATE       QPEX      QSYS
    QDFTOWN   QLPAUTO     QPM400    QTCP
    QDIRSRV   QLPINSTALL  QRJE      QTFTP
    ```

  - To obtain a profile token for a profile that is disabled, specify *NOPWDCHK for the password parameter.
  - To obtain a profile token when the password is expired, specify *NOPWDCHK or *NOPWDSTS for the password parameter.
- Generates the profile token designating the user's authorities.

  The maximum number of profile tokens that can be generated is approximately 2,000,000; after that, the space to store them is full. Message CPF4AAA is sent to the application, and no more profile tokens can be generated until one is removed.
- Updates the last-used date for the user and its group profiles.
- Resets the signon attempts not valid count to zero when a profile token is successfully generated for a user.
- If security-related events are being audited, adds an entry to the QAUDJRN audit journal to indicate that a profile token is created.

Authorities and Locks

API Public Authority
> *USE

*User profile authority, if the password is *NOPWD *NOPWDCHK or *NOPWDSTS.*
> *USE

User Profile Lock
> *LSRD

Required Parameter Group

Profile token
> OUTPUT; CHAR(32)

> The profile token that is generated.

User profile name
> INPUT; CHAR(10)

> The name of the user for which to generate the profile token.

User password
> INPUT; CHAR(*)

> The password of the user for which to generate the profile token or a special value.

> **Password of the user**

> • *Length of password* and *CCSID of password* are required

> **Special value**

> • *Length of password* and *CCSID of password* are not allowed when specifying a special value.

> • A special value must be a 10 character, blank padded value in CCSID 37.

> • Special values allowed are:

*NOPWD	The user requesting the profile token must have *USE authority to the user profile.
	A profile token does not get created for a disabled user profile.
	A profile token does not get created for a user profile with an expired password.
	This value is not allowed if the name of the currently running profile is specified for the user profile name parameter.
*NOPWDCHK	The user requesting the profile token must have *USE authority to the user profile.
	If the profile is disabled, the user requesting the profile token must have *ALLOBJ and *SECADM special authorities to get a token.
	If the password is expired, the user requesting the profile token must have *ALLOBJ and *SECADM special authorities to get a token.
*NOPWDSTS	The user requesting the profile token must have *USE authority to the user profile.
	A profile token does not get created for a disabled user profile.
	If the password is expired, the user requesting the profile token must have *ALLOBJ and *SECADM special authorities to get a token.

Time out interval
> INPUT; BINARY(4)

> The time before the profile token times out.

You can specify one of the following values:

-1   Use system default value (3600 seconds)
1-3600   Time out value in seconds.

Profile token type
      INPUT; CHAR(1)

      The type of the profile token to be generated.

      You can specify one of the following values:

1   Single-use profile token. A single-use profile token can be used only on the Set To Profile Token (QSYSETPT;
        QsySetToProfileToken) API once and cannot be used to generate new profile tokens.
2   Multiple-use profile token. A multiple-use profile token can be used on the Set To Profile Token (QSYSETPT;
        QsySetToPrfTkn) API an unlimited number of times, but cannot be used to generate new profile tokens.
3   Multiple-use, regenerable profile token. A multiple-use, regenerable profile token can be used on the Set To
        Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times and can be used to generate a
        new single-use, multiple-use, or multiple-use, regenerable profile token.

Error code
      I/O; CHAR(*)

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

Optional Parameter Group

This parameter group is required when specifying a password for the *password* parameter. It is not
allowed when specifying a special value.

Length of user password
      INPUT; BINARY(4)

      The length, in bytes, of the password contained in the user password parameter.

      The valid values are:

1-512   The length of the password in the user password parameter.

CCSID of user password
      INPUT; BINARY(4)

      The CCSID of the user password parameter. For a list of valid CCSIDs, see the Globalization
      topic in the iSeries Information Center.

      The valid values are:

-1   The current password level for the system is used to determine the CCSID of the password data. When
        calling this API on password level 0 or 1, CCSID 37 is used. When calling this API on password level 2
        or 3, the default CCSID (DFTCCSID) job attribute is used. See usage notes for more details.
0   The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is
        65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533 A valid CCSID in this range.

Usage Notes

The CCSID parameter on this API can lead to potential problems if coded with inconsistent CCSID
values. Passwords created using the CRTUSRPRF, CHGUSRPRF, and CHGPWD CL commands, as well as
the QSYCHGPW API (when called without passing the CCSID parameter), while the system is running
password level 0 or 1 are created using CCSID 37. Passwords created using these CL commands and the

QSYCHGPW API (without the CCSID parameter specified) when running password level 2 or 3 are created using the default job CCSID. Using variant characters $, @ and #, as well as other variant characters, in a user password may result in inconsistencies when converting from one CCSID to another. When calling this API on password level 0 or 1, CCSID 37 should be specified unless the password string is in a known CCSID. When calling this API on password level 2 or 3, pass the default job CCSID unless the password string is in a known CCSID.

Error Messages

Message ID	Error Message Text
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF227F E	*NOPWD not allowed for current user.
CPF22E2 E	Password not correct for user profile &1.
CPF22E3 E	User profile &1 is disabled.
CPF22E4 E	Password for user profile &1 has expired.
CPF22E5 E	No password associated with user profile &1.
CPF22E9 E	*USE authority to user profile &1 required.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3BDE E	CCSID &1 not supported by API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF4AAA E	Maximum number of profile tokens have been generated.
CPF4AAB E	Time out value not valid.
CPF4AAD E	Profile token type not valid.
CPF4AB8 E	Insufficient authority for user profile &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R5

Generate Profile Token Extended (QsyGenPrfTknE) API

»

Syntax for QsyGenPrfTknE:

```
#include <qsyptkn.h>

void QsyGenPrfTknE
     (unsigned char  *Profile_token,
      char           *User_profile_name,
      char           *User_password,
      int             Length_of_user_password,
      unsigned int    CCSID_of_user_password,
      int             Time_out_interval,
      char            Profile_token_type,
      void           *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

《

The Generate Profile Token Extended (QsyGenPrfTknE) API verifies that the caller has authority to generate a profile token for the requested profile and then generates a profile token. This profile token can be passed to one or more additional processes which can then use it to perform tasks on behalf of the authenticated user.

This API requires the password for the profile to be specified. If you need to generate a profile token for a profile without specifying the password, see the Generate Profile Token (QsyGenPrfTkn) API.

The Generate Profile Token API follows this process:
- Verifies that the user ID and password value are correct. Incorrect password values and special cases are handled as follows:

  - If the password is not correct, the incorrect password count is increased. (The QMAXSIGN system value contains the maximum number of incorrect attempts to sign on.) If the QMAXSGNACN system value is set to disable the user profile, repeated attempts to generate a profile token using an incorrect password disables the user ID. This keeps applications from methodically determining user passwords.
  - To obtain a profile token for a profile that does not have a password, use the Generate Profile Token (QsyGenPrfTkn) API.
  - To obtain a profile token for a profile that is disabled, use the Generate Profile Token (QsyGenPrfTkn) API.
  - To obtain a profile token when the password is expired, use the Generate Profile Token (QsyGenPrfTkn) API.
- Generates the profile token designating the user's authorities.

  The maximum number of profile tokens that can be generated is approximately 2,000,000; after that, the space to store them is full. Message CPF4AAA is sent to the application, and no more profile tokens can be generated until one is removed.
- Updates the last-used date for the user and its group profiles.
- Resets the signon attempts not valid count to zero when a profile token is successfully generated for a user.
- If security-related events are being audited, adds an entry to the QAUDJRN audit journal to indicate that a profile token is created.

Authorities and Locks

API Public Authority
> *USE

User Profile Lock
> *LSRD

Required Parameter Group

Profile token
> OUTPUT; CHAR(32)

> The profile token that is generated.

User profile name
> INPUT; CHAR(10)

> The name of the user for which to generate the profile token.

User password
> INPUT; CHAR(*)

> The password of the user for which to generate the profile token.

> Special values are not allowed for this parameter.

Length of user password
> INPUT; BINARY(4)

> The length, in bytes, of the password contained in the user password parameter.

> The valid values are:

1-512   The length of the password in the password parameter.

CCSID of user password
> INPUT; BINARY(4)

> The CCSID of the user password parameter. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

> The valid values are:

-1   The current password level for the system is used to determine the CCSID of the password data. When calling this API on password level 0 or 1, CCSID 37 is used. When calling this API on password level 2 or 3, the default CCSID (DFTCCSID) job attribute is used. See usage notes for more details.

0   The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.

1-65533   A valid CCSID in this range.

Time out interval
> INPUT; BINARY(4)

> The time before the profile token times out.

> You can specify one of the following values:

-1   Use system default value (3600 seconds)

1-3600   Time out value in seconds.

Profile token type
> INPUT; CHAR(1)

> The type of the profile token to be generated.

> You can specify one of the following values:

1   Single-use profile token. A single-use profile token can be used only on the Set To Profile Token (QSYSETPT; QsySetToProfileToken) API once and cannot be used to generate new profile tokens.

2   Multiple-use profile token. A multiple-use profile token can be used on the Set To Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times, but cannot be used to generate new profile tokens.

3   Multiple-use, regenerable profile token. A multiple-use, regenerable profile token can be used on the Set To Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times and can be used to generate a new single-use, multiple-use, or multiple-use, regenerable profile token.

Error code
> I/O; CHAR(*)

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

Usage Notes

The CCSID parameter on this API can lead to potential problems if coded with inconsistent CCSID values. Passwords created using the CRTUSRPRF, CHGUSRPRF, and CHGPWD CL commands, as well as the QSYCHGPW API (when called without passing the CCSID parameter), while the system is running password level 0 or 1 are created using CCSID 37. Passwords created using these CL commands and the QSYCHGPW API (without the CCSID parameter specified) when running password level 2 or 3 are created using the default job CCSID. Using variant characters $, @ and #, as well as other variant characters, in a user password may result in inconsistencies when converting from one CCSID to another. When calling this API on password level 0 or 1, CCSID 37 should be specified unless the password string is in a known CCSID. When calling this API on password level 2 or 3, pass the default job CCSID unless the password string is in a known CCSID.

Error Messages

Message ID	Error Message Text
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF227F E	*NOPWD not allowed for current user.
CPF22E2 E	Password not correct for user profile &1.
CPF22E3 E	User profile &1 is disabled.
CPF22E4 E	Password for user profile &1 has expired.
CPF22E5 E	No password associated with user profile &1.
CPF22E9 E	*USE authority to user profile &1 required.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3BDE E	CCSID &1 not supported by API.
CPF3CF1 E	Error code parameter not valid.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF4AAA E	Maximum number of profile tokens have been generated.
CPF4AAB E	Time out value not valid.
CPF4AAD E	Profile token type not valid.
CPF4AB8 E	Insufficient authority for user profile &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

Generate Profile Token From Profile Token (QSYGENFT, QsyGenPrfTknFromPrfTkn) API

≫

Required Parameter Group for QSYGENFT:

1	New profile token	Output	Char(32)
2	From profile token	Input	Char(32)
3	Time out interval	Input	Bin(4)
4	New profile token type	Input	Char(1)
5	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

Syntax for QsyGenPrfTknFromPrfTkn:

```
#include <qsyptkn.h>

void QsyGenPrfTknFromPrfTkn
      (unsigned char  *New_profile_token,
       unsigned char  *From_profile_token,
       int            *Time_out_interval,
       char           *New_profile_token_type,
       void           *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Generate Profile Token From Profile Token (OPM, QSYGENFT; ILE, QsyGenPrfTknFromPrfTkn) API generates a profile token using an existing profile token. The existing profile token must be a valid, multiple-use, regenerable profile token. The new profile token will represent the same user and group information as the original profile token.

Authorities and Locks

API Public Authority
        *USE

Required Parameter Group

Profile token
        OUTPUT; CHAR(32)

        The profile token that is generated.

From profile token
        INPUT; CHAR(32)

        The multiple-use, regenerable profile token used to generate the new profile token.

Time out interval
        INPUT; BINARY(4)

        The time in seconds before the new profile token times out.

        You can specify one of the following values:

-1   Use system default value (3600 seconds)
1-3600   Time out value in second.

New profile token type
        INPUT; CHAR(1)

        You can specify one of the following values:

1   Single-use profile token. A single-use profile token can be used only on the Set To Profile Token (QSYSETPT; QsySetToProfileToken) API once and cannot be used to generate new profile tokens.
2   Multiple-use profile token. A multiple-use profile token can be used on the Set To Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times, but cannot be used to generate new profile tokens.
3   Multiple-use, regenerable profile token. A multiple-use, regenerable profile token can be used on the Set To Profile Token (QSYSETPT; QsySetToPrfTkn) API an unlimited number of times and can be used to generate a new single-use, multiple-use, or multiple-use, regenerable profile token.

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
CPF2225 E	Not able to allocate internal system object.
CPF2274 E	Profile token not valid.
CPF229F E	Profile token not valid type.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF4AAA E	Maximum number of profile tokens have been generated.
CPF4AAB E	Time out value not valid.
CPF4AAD E	Profile token type not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R5

Get Profile Handle (QSYGETPH) API

Required Parameter Group:

1	User ID	Input	Char(10)
2	Password	Input	Char(*)
3	Profile handle	Output	Char(12)

Optional Parameter Group 1:

4	Error code	I/O	Char(*)

Optional Parameter Group 2:

5	Length of password	Input	Bin(4)
6	CCSID of password	Input	Bin(4)

Default Public Authority: *USE
Threadsafe: Yes

The Get Profile Handle (QSYGETPH) API validates user IDs and passwords and creates a profile handle
for use in jobs that run under more than one user profile. The profile handle is temporary; you can use it
only in the job that created it.

The QSYGETPH API follows this process:

- Verifies that the user ID and password are correct. Incorrect passwords and special cases are handled
  as follows:

  - If the password is not correct, the incorrect password count is increased. (The QMAXSIGN system
    value contains the maximum number of incorrect attempts to sign on.) If the QMAXSGNACN
    system value is set to disable the user profile, repeated attempts to validate an incorrect password
    disable the user ID. This keeps applications from methodically determining user passwords.

- To obtain a profile handle for a profile that does not have a password, specify *NOPWD, *NOPWDCHK or *NOPWDSTS for the password parameter.

  You cannot obtain a profile handle for the following system-supplied user profiles:

  ```
  QAUTPROF   QDLFM       QMSF       QSNADS     QTSTRQS
  QCLUMGT    QDOC        QNETSPLF   QSPL
  QCOLSRV    QDSNX       QNFSANON   QSPLJOB
  QDBSHR     QFNC        QNTP       QSRVAGT
  QDBSHRDO   QGATE       QPEX       QSYS
  QDFTOWN    QLPAUTO     QPM400     QTCP
  QDIRSRV    QLPINSTALL  QRJE       QTFTP
  ```

- To obtain a profile handle for a profile that is disabled, specify *NOPWDCHK for the password parameter.
- To obtain a profile handle when the password is expired, specify *NOPWDCHK or *NOPWDSTS for the password parameter.
- Generates the profile handle, a 12-character random string designating the user's authorities. This string, not the user's password, supplies the Set Profile Handle (QWTSETP, QsySetProfileHandle) and the Release Profile Handle (QSYRLSPH, QsyReleaseHandle) APIs.

  The maximum number of profile handles that can be created is approximately 20,000 per job; after that, the space to store them is full. Message CPF22E6 is sent to the application, and QSYGETPH stops generating profile handles.

  Be sure to keep track of the profile handles created in the calling application. If the application calls QSYGETPH twice with the same user profile and password, QSYGETPH returns two different profile handles. Either handle can be used, but generating and using just one is more efficient.
- Updates the last-used date for the user and group profiles.
- Resets the signon attempts not valid count to zero.
- If security-related events are being audited, adds an entry to the QAUDJRN audit journal to indicate that a profile handle is created.

Authorities and Locks

API Public Authority
> *USE

*User profile authority, if the password is *NOPWD, *NOPWDCHK or *NOPWDSTS.*
> *USE

User Profile Lock
> *LSRD

Required Parameter Group

User ID
> INPUT; CHAR(10)
>
> The user ID of the profile for which the handle is being created. A user ID must be a 10 character, blank padded value in CCSID 37.
>
> You can specify the following special value:

*CURRENT   A handle is generated with the current thread information.

> When specifying *CURRENT, *password* is ignored and *length of password* and *CCSID of password* are not allowed.

Password
> INPUT; CHAR(*)
>
> The password for the user ID or a special value.

Password for the user ID

- *Length of password* and *CCSID of password* are required

Special value

- *Length of password* and *CCSID of password* are not allowed when specifying a special value.
- A special value must be a 10 character, blank padded value in CCSID 37.
- Special values allowed are:

NOPWD	The user requesting the profile handle must have *USE authority to the user profile.
	A profile handle does not get created for a disabled user profile.
	A profile handle does not get created for a user profile with an expired password.
NOPWDCHK	The user requesting the profile handle must have *USE authority to the user profile.
	If the profile is disabled, the user requesting the profile handle must have *ALLOBJ and *SECADM special authorities to get a handle.
	If the password is expired, the user requesting the profile handle must have *ALLOBJ and *SECADM special authorities to get a handle.
NOPWDSTS	The user requesting the profile handle must have *USE authority to the user profile.
	A profile handle does not get created for a disabled user profile.
	If the password is expired, the user requesting the profile handle must have *ALLOBJ and *SECADM special authorities to get a handle.

Profile handle
>OUTPUT; CHAR(12)
>
>A unique string or handle designating the user profile to use as input to other routines. The handle is temporary; you can use it only in the job that created it.

Optional Parameter Group 1

This parameter group is required when specifying a password for the *password* parameter. It is optional when specifying a special value.

Error code
>I/O; CHAR(*)
>
>The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Optional Parameter Group 2

This parameter group is required when specifying a password for the *password* parameter. It is not allowed when specifying a special value.

Length of password
>INPUT; BINARY(4)
>
>The length, in bytes, of the password contained in the user profile password parameter.
>
>The valid values are:

1-512	The length of the password in the password parameter.

CCSID of password
>INPUT; BINARY(4)

The CCSID of the password parameter. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

The valid values are:

-1	The current password level for the system is used to determine the CCSID of the password data. When calling this API on password level 0 or 1, CCSID 37 is used. When calling this API on password level 2 or 3, the default CCSID (DFTCCSID) job attribute is used. See usage notes for more details.
0	The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533	A valid CCSID in this range.

Usage Notes

Profile handles are a limited resource; it is possible to run out of handles. To guarantee that you always have a profile handle to switch back to, it is recommended that you get a profile handle for both the current thread and the user profile to which you plan to switch. If for some reason you cannot do this, and if you cannot get a profile handle that will allow you to switch back, then it probably is safest to end the thread or job.

The CCSID parameter on this API can lead to potential problems if coded with inconsistent CCSID values. Passwords created using the CRTUSRPRF, CHGUSRPRF, and CHGPWD CL commands, as well as the QSYCHGPW API (when called without passing the CCSID parameter), while the system is running password level 0 or 1 are created using CCSID 37. Passwords created using these CL commands and the QSYCHGPW API (without the CCSID parameter specified) when running password level 2 or 3 are created using the default job CCSID. Using variant characters $, @ and #, as well as other variant characters, in a user password may result in inconsistencies when converting from one CCSID to another. When calling this API on password level 0 or 1, CCSID 37 should be specified unless the password string is in a known CCSID. When calling this API on password level 2 or 3, pass the default job CCSID unless the password string is in a known CCSID.

Error Messages

Message ID	Error Message Text
CPF2203 E	User profile &1 not correct.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF22E2 E	Password not correct for user profile &1.
CPF22E3 E	User profile &1 is disabled.
CPF22E4 E	Password for user profile &1 has expired.
CPF22E5 E	No password associated with user profile &1.
CPF22E6 E	Maximum number of profile handles have been generated.
CPF22E9 E	*USE authority to user profile &1 required.
CPF24B4 E	Severe error while addressing parameter list.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3BDE E	CCSID &1 not supported by API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF4AB8 E	Insufficient authority for user profile &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Get Profile Handle (QsyGetProfileHandle) API

≫

Syntax for QsyGetProfileHandle:

```
#include <qsyphandle.h>

void QsyGetProfileHandle
      (unsigned char  *Profile_handle,
       char           *User_ID,
       char           *Password,
       int             Length_of_password,
       unsigned int    CCSID_of_password,
       void           *Error_code);
```

Service Program: QSYPHANDLE
Default Public Authority: *USE
Threadsafe: Yes

≪

The Get Profile Handle (QsyGetProfileHandle) API validates user IDs and passwords and creates a profile handle, for use in jobs that run under more than one user profile. The profile handle is temporary; you can use it only in the job that created it.

This API requires the password for the user ID to be specified. If you need to create a profile handle for a user ID without specifying the password, see the Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API.

The Get Profile Handle API follows this process:

- Verifies that the user ID and password are correct. Incorrect passwords and special cases are handled as follows:

  - If the password is not correct, the incorrect password count is increased. (The QMAXSIGN system value contains the maximum number of incorrect attempts to sign on.) If the QMAXSGNACN system value is set to disable the user profile, repeated attempts to validate an incorrect password disables the user ID. This keeps applications from methodically determining user passwords.
  - To obtain a profile handle for *CURRENT user, use the Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API.
  - To obtain a profile handle for a profile that does not have a password, use the Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API.
- To obtain a profile handle for a profile that is disabled, use the Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API.
- To obtain a profile handle when the password is expired, use the Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API.
- Generates the profile handle, a 12-character random string designating the user's authorities. This string, not the user's password, supplies the Set Profile Handle (QWTSETP, QsySetProfileHandle) and the Release Profile Handle (QSYRLSPH, QsyReleaseHandle) APIs.

  The maximum number of profile handles that can be created is approximately 20,000 per job; after that, the space to store them is full. Message CPF22E6 is sent to the application, and Get Profile Handle stops generating profile handles.

Be sure to keep track of the profile handles created in the calling application. If the application calls Get Profile Handle twice with the same user profile and password, Get Profile Handle returns two different profile handles. Either handle can be used, but generating and using just one is more efficient.

- Updates the last-used date for the user and group profiles.
- Resets the signon attempts not valid count to zero.
- If security-related events are being audited, adds an entry to the QAUDJRN audit journal to indicate that a profile handle is created.

Authorities and Locks

API Public Authority
    *USE

User Profile Lock
    *LSRD

Required Parameter Group

Profile handle
    OUTPUT; CHAR(12)

    A unique string or handle designating the user profile to use as input to other routines. The handle is temporary; you can use it only in the job that created it.

User ID
    INPUT; CHAR(10)

    The user ID of the profile for which the handle is being created. A user ID must be a 10 character, blank padded value in CCSID 37.

Password
    INPUT; CHAR(*)

    The password for the user ID.

    Special values are not allowed for this parameter.

Length of password
    INPUT; BINARY(4)

    The length, in bytes, of the password contained in the user profile password parameter.

    The valid values are:

1-512   The length of the password in the password parameter.

CCSID of password
    INPUT; BINARY(4)

    The CCSID of the password parameter. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

    The valid values are:

-1   The current password level for the system is used to determine the CCSID of the password data. When calling this API on password level 0 or 1, CCSID 37 is used. When calling this API on password level 2 or 3, the default CCSID (DFTCCSID) job attribute is used. See usage notes for more details.
0   The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.
1-65533   A valid CCSID in this range.

Error code
> I/O; CHAR(*)

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

Usage Notes

Profile handles are a limited resource; it is possible to run out of handles. To guarantee that you always
have a profile handle to switch back to, it is recommended that you get a profile handle for both the
current thread and the user profile to which you plan to switch. If for some reason you cannot do this,
and if you cannot get a profile handle that will allow you to switch back, then it probably is safest to end
the thread or job.

The CCSID parameter on this API can lead to potential problems if coded with inconsistent CCSID
values. Passwords created using the CRTUSRPRF, CHGUSRPRF, and CHGPWD CL commands, as well as
the QSYCHGPW API (when called without passing the CCSID parameter), while the system is running
password level 0 or 1 are created using CCSID 37. Passwords created using these CL commands and the
QSYCHGPW API (without the CCSID parameter specified) when running password level 2 or 3 are
created using the default job CCSID. Using variant characters $, @ and #, as well as other variant
characters, in a user password may result in inconsistencies when converting from one CCSID to another.
When calling this API on password level 0 or 1, CCSID 37 should be specified unless the password string
is in a known CCSID. When calling this API on password level 2 or 3, pass the default job CCSID unless
the password string is in a known CCSID.

Error Messages

Message ID	Error Message Text
CPF2203 E	User profile &1 not correct.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF22E2 E	Password not correct for user profile &1.
CPF22E3 E	User profile &1 is disabled.
CPF22E4 E	Password for user profile &1 has expired.
CPF22E5 E	No password associated with user profile &1.
CPF22E6 E	Maximum number of profile handles have been generated.
CPF22E9 E	*USE authority to user profile &1 required.
CPF3BC7 E	CCSID &1 outside of valid range.
CPF3BDE E	CCSID &1 not supported by API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF4AB8 E	Insufficient authority for user profile &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API

≫

> Syntax for QsyGetProfileHandleNoPwd:

```
#include <qsyphandle.h>

void QsyGetProfileHandleNoPwd
      (unsigned char  *Profile_handle,
       char           *User_ID,
       char           *Password_value,
       void           *Error_code);
```

Service Program: QSYPHANDLE
Default Public Authority: *USE
Threadsafe: Yes

≪

The Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API validates user IDs and creates a profile handle, for use in jobs that run under more than one user profile. The profile handle is temporary; you can use it only in the job that created it.

This API requires a special value to be specified for the password value parameter. If you need to validate a user password, see the Get Profile Handle (QsyGetProfileHandle) API.

The Get Profile Handle No Password API follows this process:

- Verifies that the user ID and password value are correct. Incorrect password values and special cases are handled as follows:

    - To obtain a profile handle for a profile that does not have a password, specify *NOPWD, *NOPWDCHK or *NOPWDSTS for the password parameter.

      You cannot obtain a profile handle for the following system-supplied user profiles:

QAUTPROF	QDLFM	QMSF	QSNADS	QTSTRQS
QCLUMGT	QDOC	QNETSPLF	QSPL	
QCOLSRV	QDSNX	QNFSANON	QSPLJOB	
QDBSHR	QFNC	QNTP	QSRVAGT	
QDBSHRDO	QGATE	QPEX	QSYS	
QDFTOWN	QLPAUTO	QPM400	QTCP	
QDIRSRV	QLPINSTALL	QRJE	QTFTP	

    - To obtain a profile handle for a profile that is disabled, specify *NOPWDCHK for the password parameter.
    - To obtain a profile handle when the password is expired, specify *NOPWDCHK or *NOPWDSTS for the password parameter.
- Generates the profile handle, a 12-character random string designating the user's authorities. This string, not the user's password, supplies the Set Profile Handle (QWTSETP, QsySetProfileHandle) and the Release Profile Handle (QSYRLSPH, QsyReleaseHandle) APIs.

  The maximum number of profile handles that can be created is approximately 20,000 per job; after that, the space to store them is full. Message CPF22E6 is sent to the application, and Get Profile Handle stops generating profile handles.

  Be sure to keep track of the profile handles created in the calling application. If the application calls Get Profile Handle twice with the same user profile and password, Get Profile Handle returns two different profile handles. Either handle can be used, but generating and using just one is more efficient.
- Updates the last-used date for the user and group profiles.
- Resets the signon attempts not valid count to zero.
- If security-related events are being audited, adds an entry to the QAUDJRN audit journal to indicate that a profile handle is created.

Authorities and Locks

API Public Authority
> *USE

User profile authority
> *USE

User Profile Lock
> *LSRD

Required Parameter Group

Profile handle
> OUTPUT; CHAR(12)

> A unique string or handle designating the user profile to use as input to other routines. The handle is temporary; you can use it only in the job that created it.

User ID
> INPUT; CHAR(10)

> The user ID of the profile for which the handle is being created. A user ID must be a 10 character, blank padded value in CCSID 37.

> You can specify the following special value:

*CURRENT   A handle is generated with the current thread information.

Password value
> INPUT; CHAR(10)

> The password value for the user ID.

> Only special values are allowed for this parameter. A special value must be a 10 character, blank padded value in CCSID 37.

> You must specify one of the following special values:

*NOPWD   The user requesting the profile handle must have *USE authority to the user profile.

                A profile handle does not get created for a disabled user profile.

                A profile handle does not get created for a user profile with an expired password.

*NOPWDCHK   The user requesting the profile handle must have *USE authority to the user profile.

                If the profile is disabled, the user requesting the profile handle must have *ALLOBJ and *SECADM special authorities to get a handle.

                If the password is expired, the user requesting the profile handle must have *ALLOBJ and *SECADM special authorities to get a handle.

*NOPWDSTS   The user requesting the profile handle must have *USE authority to the user profile.

                A profile handle does not get created for a disabled user profile.

                If the password is expired, the user requesting the profile handle must have *ALLOBJ and *SECADM special authorities to get a handle.

Error code
> I/O; CHAR(*)

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

Usage Notes

Profile handles are a limited resource. It is possible to run out. Therefore, to guarantee that you will always have a profile handle to switch back to, it is recommended that you get a profile handle for both the current thread and the user profile you plan to switch to. If for some reason you cannot do this, and if you cannot get a profile handle that will allow you to switch back then it is probably safest to just end the thread or job.

Error Messages

Message ID	Error Message Text
CPF2203 E	User profile &1 not correct.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF22E3 E	User profile &1 is disabled.
CPF22E4 E	Password for user profile &1 has expired.
CPF22E6 E	Maximum number of profile handles have been generated.
CPF22E9 E	*USE authority to user profile &1 required.
CPF3C90 E	Literal value cannot be changed.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF4AB8 E	Insufficient authority for user profile &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R3

Get Profile Token Time Out (QSYGETPT, QsyGetPrfTknTimeOut) API

≫

Required Parameter Group for QSYGETPT:

1	Time out	Output	Binary(4)
2	Profile token	Input	Char(32)
3	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

Syntax for QsyGetPrfTknTimeOut:

```
#include <qsyptkn.h>

void QsyGetPrfTknTimeOut
      (int           *Time_out,
       unsigned char *Profile_token,
       void          *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Get Profile Token Time Out (OPM, QSYGETPT; ILE, QsyGetPrfTknTimeOut) API gets the number of seconds until a profile token is not valid.

Authorities and Locks

API Public Authority
    *USE

Required Parameter Group

Time out
    OUTPUT; BINARY(4)

    The seconds until the profile token times out. If 0 is returned, the profile token is no longer valid.

Profile token
    Input; CHAR(32)

    The profile token for which to get the time out.

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
CPF2225 E	Not able to allocate internal system object.
CPF2274 E	Profile token is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R5

Invalidate Profile Token (QSYINVPT, QsyInvalidatePrfTkn) API

≫

Required Parameter Group for QSYINVPT:

1	Profile token	Input	Char(32)
2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

Syntax for QsyInvalidatePrfTkn:

```
#include <qsyptkn.h>

void QsyInvalidatePrfTkn
      (unsigned char  *Profile_token,
       void           *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

«

The Invalidate Profile Token (OPM, QSYINVPT; ILE, QsyInvalidatePrfTkn) API invalidates a profile
token. The profile token is no longer usable for other profile token APIs except the Remove Profile Token
(QSYRMVPT, QsyRemovePrfTkn) API.

Authorities and Locks

API Public Authority
    *USE

Required Parameter Group

Profile token
    INPUT; CHAR(32)

    The profile token to be invalidated.

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
PF2225 E	Not able to allocate internal system object.
CPF2274 E	Profile token is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R5

List Authorized Users (QSYLAUTU) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The List Authorized Users (QSYLAUTU) API puts a list of authorized system users into a user space.

This API provides information similar to the Display Authorized Users (DSPAUTUSR) command.

Authorities and Locks

User Space Authority
> *CHANGE

Authority to Library Containing User Space
> *EXECUTE

Authority to User Profiles in List of Authorized Users
> *READ, only those profiles that you have *READ authority to are returned.

Required Parameter Group

Qualified user space name
> INPUT; CHAR(20)
>
> The name of the existing user space that the list of authorized users is returned to. The first 10 characters specify the user space name, and the second 10 characters specify the library.
>
> You can use these special values for the library name:

*CURLIB	The current library is searched for the user space. If there is no current library, QGPL (general purpose library) is used.
*LIBL	The library list is searched for the user space.

Format name
> INPUT; CHAR(8)
>
> The name of the format used to list the authorized users.
>
> You can specify these formats:

"AUTU0100 Format" on page 52	Each entry contains the user name, group names, an indicator that specifies whether the user is a user profile or a group profile, and an indicator that specifies whether the user is a group that has members.
"AUTU0200 Format" on page 52	Each entry contains the same information as AUTU0100 plus the text description for the user.

Error code
> I/O; CHAR(*)
>
> The structure in which to return error information. For the format of the structure, see Error Code Parameter.

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 52.

Input Parameter Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name
10	0A	CHAR(10)	User space library name

Offset			
Dec	**Hex**	**Type**	**Field**
20	14	CHAR(8)	Format name

AUTU0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User profile name
10	0A	CHAR(10)	Group name
20	14	BINARY(4)	Number of supplemental groups
24	18	ARRAY(15) OF CHAR(10)	Supplemental groups
174	AE	CHAR(1)	User or group indicator
175	AF	CHAR(1)	Group member indicator

AUTU0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User profile name
10	0A	CHAR(10)	Group name
20	14	CHAR(50)	Text name
70	46	CHAR(2)	Reserved
72	48	BINARY(4)	Number of supplemental groups
76	4C	ARRAY(15) OF CHAR(10)	Supplemental groups
226	E2	CHAR(1)	User or group indicator
227	E3	CHAR(1)	Group member indicator

Field Descriptions

Format name. The name of the format used to list authorized users.

Group member indicator. Whether this user is a group that has members.

Possible values follow:

0	The user is not a group, or is a group but does not have any members. This value is returned if the user or group indicator field is 0.
1	The user is a group that has members.

Group name. The name of the user's group profile. If the user does not have a group profile, this field contains *NONE.

Number of supplemental groups. The number of supplemental groups returned in the supplemental groups field. The number of supplemental groups will be zero if the user does not have any supplemental groups.

Reserved. An ignored field.

Supplemental groups. The array of supplemental groups for the user profile. The number of supplemental groups field will indicate how many entries there are in the array.

Text name. The text description for the authorized user.

User profile name. The name of the authorized user.

User space name. The name of the user space used to return the list of authorized users on the system.

User space library name. The name of the library containing the user space.

User or group indicator. Whether this user is a user profile or a group profile.

Possible values follow:

0   User profile (profile does not have a GID)
1   Group profile (profile has a GID)

Error Messages

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R4

Retrieve Authorization List Information (QSYRTVAI) API

Required Parameter Group:

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

4	Authorization list name	Input	Char(10)
5	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Authorization List Information (QSYRTVAI) API returns to the caller information about the number of authorization list entries that are in use for the specified authorization list. The returned information can be use to determine how full an authorization list is.

The normal view of an authorization list (AUTL) is that it is a single object (structure) with the capacity to secure 2,097,104 objects. However, to successfully manage the fullness of an AUTL, a more comprehensive view of an AUTL is required.

When an object is secured by an AUTL, each piece of the object may have to be individually secured by the AUTL. This means that when an object is secured by an AUTL, multiple entries in the AUTL may be used by the system to secure all of the parts of the object. For example, when a database file with multiple members is secured by an AUTL, multiple entries in the AUTL will be used. Because of this, using commands like DSPAUTLOBJ (Display Authorization List Objects) and APIs like QSYLATLO (List Authorization List Objects), is not an effective way to determine whether an AUTL is becoming full.

In addition, the first time an AUTL is used to secure an object that resides in an independent auxiliary storage pool (ASP), the system creates an internal structure in the independent ASP that is transparently tied to the AUTL object. This internal structure is called an authorization list extension. The authorization list extension has entries just like the AUTL but these entries can only be used to secure objects that reside in the independent ASP. An authorization list extension also has a maximum of 2,097,104 entries. These entries are in addition to the 2,097,104 entries in the base AUTL which are used to secure objects in the system and basic user auxiliary storage pools (*SYSBAS). The system can have a maximum of 223 independent ASPs; therefore, the theoretical combined limit for an AUTL and its extensions is 469,751,296 entries (224 multiplied by 2,097,104).

If an attempt is made to secure an object that resides in an independent ASP with an AUTL but all of the entries in the authorization list extension for the object's independent ASP have been used; an MCH2804 exception will occur indicating that the storage limit for the AUTL has been reached. This would happen even if there were unused entries available for objects that reside in *SYSBAS. Likewise, if an attempt is made to secure an object that resides in *SYSBAS with an AUTL but all of the AUTL's *SYSBAS entries have been used; an MCH2804 exception will occur even if there are unused entries available in authorization list extensions for the AUTL.

To effectively manage the fullness of an AUTL, a user needs to ensure that no single set of entries (the *SYSBAS set and the authorization list extension sets) exceed their individual 2,097,104 entry limit.

This API returns information about the total number of entries used for *SYSBAS and all varied on independent ASPs for the specified AUTL as well as detailed entry use information for *SYSBAS and each varied on independent ASP.

Authorities and Locks

Authorization List Authority
      *READ

Required Parameter Group

Receiver variable
      OUTPUT; CHAR(*)

The variable used to return the information about the AUTL. An approximation of the size needed for this receiver variable can be obtained by calling the QSYRTVAI API with the *length of receiver variable* set to 8 bytes. The *bytes available* value that is returned in this receiver variable will indicate the approximate size needed for the receiver variable. The receiver variable format is defined in "RTAI0100 Format."

Length of receiver variable
    INPUT; BINARY(4)

The length of the receiver variable. This value must be at least 8.

Format
    INPUT; CHAR(8)

The name of the format that is used to return the AUTL's information.

The following value is allowed:

"RTAI0100 Format"   Authorization list information is returned.

Authorization list name
    INPUT; CHAR(10)

The name of the authorization list for which information is to be returned. The name must be specified in uppercase.

Error code
    I/O; CHAR(*)

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

Format of Receiver Variable

The following tables describe the receiver variable that is returned by the QSYRTVAI API.

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 56.

RTAI0100 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(8)	Total entries used
16	10	BINARY(4)	Offset to ASP information sets
20	14	BINARY(4)	Number of ASP information sets
24	18	BINARY(4)	Length of ASP information set entry
*	*	Array of CHAR(*)	ASP information sets

ASP Information Set

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	ASP name
10	A	CHAR(1)	AUTL extension indicator

Offset			
Dec	Hex	Type	Field
11	B	CHAR(1)	Reserved
12	C	BINARY(4)	Number of entries used
16	10	BINARY(4)	Number entries available

Field Descriptions

ASP name. The name of the ASP whose information is being returned. The system ASP and basic user ASP information is returned with an ASP name of ***SYSBAS**. For independent ASPs, the ASP name is the name of the varied on device description associated with the independent ASP.

AUTL extension indicator. This field indicates whether an authorization list extension exists for the specified AUTL on the varied on independent ASP. The possible values for this field are:

Blank This is the value returned for *SYSBAS. AUTL extensions are not applicable to the base AUTL object that secures objects in *SYSBAS.

0 An AUTL extension for the specified AUTL does not exist on this independent ASP.

1 An AUTL extension for the specified AUTL does exist on this independent ASP.

Bytes available. The number of bytes of data available to be returned to the user.

If the *bytes available* field is larger than the *bytes returned* field, all of the ASP information will not be included in the receiver variable.

Bytes returned. The number of bytes of data returned to the user in the receiver variable.

Length of ASP information set entry. The length of a single ASP information set.

Number of ASP information sets. The number of ASP information sets. This is the total number of information sets that could be returned. There is always one ASP information set for *SYSBAS. Additional ASP information sets are provided for each varied on independent ASP even if the authorization list does not currently secure any objects in the independent ASP. The ASP information set for *SYSBAS is always the first ASP information set returned.

Number of entries used. The number of entries used for securing objects that reside in the ASP. This field can have a value from zero to 2,097,104.

Number entries available. The number of entries available for securing additional objects that reside in the ASP. This field can have a value from zero to 2,097,104.

Offset to ASP information sets. The offset from the beginning of the receiver variable to the begining of the ASP information sets.

Reserved. Reserved space used for boundary alignment.

Total entries used. The total number of authorization list entries used for all objects secured by the authorization list that reside in the system ASP, basic user ASPs, and all varied on independent ASPs. This field can have a value ranging from zero to the (number-of-varied-on-independent-ASPs + 1) multiplied by 2,097,104.

Error Messages

Message ID	Error Message Text
CPF22F0 E	Unexpected errors occurred during processing.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of receiver variable is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF8132 E	Authorization list &4 damaged.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R4

List Objects Secured by Authorization List (QSYLATLO) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Authorization list	Input	Char(10)
4	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The List Objects Secured by Authorization List (QSYLATLO) API puts a list of objects secured by an authorization list into a user space.

This API provides information similar to the Display Authorization List Objects (DSPAUTLOBJ) command.

Authorities and Locks

User Space Authority
        *CHANGE

Authority to Library Containing User Space
        *EXECUTE

Authorization List Authority
        Must not be *EXCLUDE authority

Required Parameter Group

Qualified user space name
        INPUT; CHAR(20)

        The name of the existing user space where the list of objects secured by the authorization list is returned to. The first 10 characters specify the user space name, and the second 10 characters specify the library.

You can use these special values for the library name:

CURLIB   The current library is used to locate the user space. If there is no current library, QGPL (general purpose library)

LIBL   The library list is used to locate the user space.

Format name
INPUT; CHAR(8)

The name of the format used to list objects secured by the authorization list.

You can specify these formats:

"ATLO0100 Format" on page 59	Each entry contains the object name, library, type, authority holder indicator, auxiliary storage pool (ASP) device name of library, and ASP device name of object.
"ATLO0110 Format" on page 60	This format only returns path names for objects in a directory. Each entry contains the offset to the path name, the length of the path name, type, authority holder indicator, ASP device name of object, and the path name value. Objects in the QSYS.LIB and QDLS file systems are not returned with this format.
"ATLO0200 Format" on page 60	Each entry contains the same information as ATLO0100 plus the object owner, attribute, text, and primary group.
"ATLO0210 Format" on page 60	This format only returns path names for objects in a directory. Each entry contains the same information as format ATLO0110 plus the object owner, attribute, text, and primary group. Objects in the QSYS.LIB and QDLS file systems are not returned with this format.
"ATLO0300 Format" on page 61	Each entry contains the length of the entry, object name, library, type, authority holder indicator, document library object (DLO) name, the name of the folder that the DLO is in, the displacement to the path name, the length of the path name, ASP device name of library, ASP device name of object, and the path name value. Objects in all file systems are returned with this format. Objects are returned consecutively in three groups. Objects in the QSYS.LIB file system are in one group, objects in the QDLS file system are in another group, and objects in directories are in the other group. Information returned in the Header Section of the user space indicates how to get to the beginning of each group of objects.
"ATLO0400 Format" on page 61	Each entry contains the same information as ATLO0300 plus the object owner, primary group, attribute, and text. Objects in all file systems are returned with this format. Objects are returned consecutively in three groups. Objects in the QSYS.LIB file system are in one group, objects in the QDLS file system are in another group, and objects in directories are in the other group. Information returned in the Header Section of the user space indicates how to get to the beginning of each group of objects.

Authorization list
INPUT; CHAR(10)

The name of the authorization list for which the secured objects are returned.

Error code
I/O; CHAR(*)

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

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 62. When you retrieve list entry information for formats ATLO0100, ATLO0200, ATLO0300, or ATLO0400 you must use the entry size returned in the generic header to access list entries as the size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. When you retrieve list entry information for

formats ATLO0110 and ATLO0210 you must use the **offset to path name** of the current entry + the **length of path name** of the current entry to access the next entry.

Input Parameter Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name specified
10	0A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	Authorization list

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Authorization list
10	0A	CHAR(10)	Authorization list library name
20	14	CHAR(10)	Owner
30	1E	CHAR(10)	Primary group
40	28	BINARY(4)	Reason code
44	2C	BINARY(4)	Offset to first QSYS.LIB object
48	30	BINARY(4)	Entry number of first QSYS.LIB object
52	34	BINARY(4)	Number of QSYS.LIB objects
56	38	BINARY(4)	Offset to first QDLS object
60	3C	BINARY(4)	Entry number of first QDLS object
64	40	BINARY(4)	Number of QDLS objects
68	44	BINARY(4)	Offset to first directory object
72	48	BINARY(4)	Entry number of first directory object
76	4C	BINARY(4)	Number of directory objects

ATLO0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(10)	ASP device name of library
41	29	CHAR(10)	ASP device name of object

ATLO0110 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(1)	Reserved
20	14	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

ATLO0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(10)	Owner
41	29	CHAR(10)	Attribute
51	33	CHAR(50)	Text description
101	65	CHAR(10)	Primary group
111	6F	CHAR(10)	ASP device name of library
121	79	CHAR(10)	ASP device name of object

ATLO0210 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(10)	Owner
29	1D	CHAR(10)	Attribute
39	27	CHAR(50)	Text description
89	59	CHAR(10)	Primary group
99	63	CHAR(1)	Reserved
100	64	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

ATLO0300 Format

Offset Dec	Offset Hex	Type	Field
0	0	CHAR(10)	Length of entry
4	4	CHAR(10)	Object name
14	0E	CHAR(10)	Library name
24	18	CHAR(10)	Object type
34	22	CHAR(1)	Authority holder
35	23	CHAR(12)	DLO name
47	2F	CHAR(63)	Folder name
110	6E	CHAR(2)	Reserved
112	70	BINARY(4)	Displacement to path name
116	74	BINARY(4)	Length of path name
120	78	CHAR(10)	ASP device name of library
130	82	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

ATLO0400 Format

Offset Dec	Offset Hex	Type	Field
0	0	CHAR(10)	Length of entry
4	4	CHAR(10)	Object name
14	0E	CHAR(10)	Library name
24	18	CHAR(10)	Object type
34	22	CHAR(1)	Authority holder
35	23	CHAR(12)	DLO name
47	2F	CHAR(63)	Folder name
110	6E	CHAR(2)	Reserved
112	70	BINARY(4)	Displacement to path name
116	74	BINARY(4)	Length of path name
120	78	CHAR(10)	Owner
130	82	CHAR(10)	Attribute
140	8C	CHAR(50)	Text description
190	BE	CHAR(10)	Primary group
200	C8	CHAR(10)	ASP device name of library
210	D2	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

Field Descriptions

ASP device name of library. The auxiliary storage pool (ASP) device name where the object's library is stored. If the object's library is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

ASP device name of object. The auxiliary storage pool (ASP) device name where the object is stored. If the object is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

Attribute. The attribute of the secured object. If the object is not in the QSYS.LIB or QDLS file system, this field is blank.

Authority holder. Whether the object is an authority holder. If the object is an authority holder, this field is Y. If not, this field is N.

Authorization list. The name of the authorization list for which the list of objects is returned.

Authorization list library name. The name of the library containing the authorization list.

Displacement to path name. The displacement in the entry to the start of the path name.

DLO name. The document library object (DLO) name for the object. If the object is not an *DOC (document) or *FLR (folder) object, this field is blank.

Entry number of first directory object. The entry number of the first directory object (objects not in the QSYS.LIB or QDLS file system) that is returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of directory objects field is 0, this value is also 0.

Entry number of first QDLS object. The entry number of the first QDLS object that is returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of QDLS objects field is 0, this value is also 0.

Entry number of first QSYS.LIB object. The entry number of the first QSYS.LIB object that is returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of QSYS.LIB objects field is 0, this value is also 0.

Folder name. The name of the folder that contains the DLO object. If the object is not in a folder, this field contains *NONE.

Format name. The name of the format that is used to list objects secured by the authorization list.

Length of entry. The length (in bytes) of the current entry.

Length of path name. The length (in bytes) of the path name.

Library name. The name of the library that contains the user space, object, or authorization list.

Number of directory objects. The number of objects in directories (objects not in the QSYS.LIB or QDLS file system) that are returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If there are no entries for objects in directories in the user space, 0 is returned.

Number of QDLS objects. The number of objects in the QDLS file system that were returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If there are no entries for QDLS objects in the user space, 0 is returned.

Number of QSYS.LIB objects. The number of objects in the QSYS.LIB file system that were returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If there are no entries for QSYS.LIB objects in the user space, 0 is returned.

Object name. The name of the object secured by the authorization list. If the object is not in the QSYS.LIB or QDLS file system, this field is blank.

Object type. The type of secured object.

Offset to first directory object. The offset to the first directory object (objects not in the QSYS.LIB or QDLS file systems) that was returned in the user space. This value is only set if using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If 'Number of directory objects' is 0, this value will also be 0.

Offset to first QDLS object. The offset to the first QDLS object that is returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of QDLS objects field is 0, this value is also 0.

Offset to first QSYS.LIB object. The offset to the first QSYS.LIB object that is returned in the user space. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of QSYS.LIB objects field is 0, this value is also 0.

Offset to path name. The offset in the user space to the start of the path name.

Owner. The name of the owner of the authorization list or object.

Path name. The path name of the object secured by the authorization list. The user must request a format that supports path names if path names are to be included in the information returned in the user space.

The structure of the path name returned is:

Description	Type
CCSID of the returned path name	Binary(4)
Country or region ID	Char(2)
Language ID	Char(3)
Reserved field	Char(3)
Flag byte	Binary(4)
Number of bytes in the path name	Binary(4)
Path delimiter	Char(2)
Reserved field	Char(10)
Path name value	Char(*)

Primary group. The name of the user who is the primary group for the authorization list or object. If there is no primary group for the authorization list or object, this field will contain a value of *NONE.

Reason code. The reason code that further describes why the list is only a subset of all objects. The following values can be returned:

- Reason code 0000. The list returned in the user space contains all objects meeting the search criteria.
- Reason code 0001. Objects were found that meet the search criteria but could not be included in the returned list. The requested format could not handle path names for directory objects.
- Reason code 0002. Objects were found that meet the search criteria but could not be included in the returned list. The requested format could not handle objects found in library QSYS.
- Reason code 0003. Directory objects were found but did not have links to them.

Reserved. This field is not used.

Text description. The descriptive text for the secured object. If the object is not in the QSYS.LIB or QDLS file system, this field is blank.

User space library name specified. The name of the library containing the user space or object.

User space name specified. The user space used to return the list of objects secured by the authorization list.

Error Messages

Message ID	Error Message Text
CPF22AF E	Not authorized to authorization list &1.
CPF2283 E	Authorization list &1 does not exist.
CPF2289 E	Unable to allocate authorization list &1.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

List Objects That Adopt Owner Authority (QSYLOBJP) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	User profile name	Input	Char(10)
4	Object type	Input	Char(10)
5	Continuation handle	Input	Char(20)
6	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The List Objects That Adopt Owner Authority (QSYLOBJP) API puts a list of objects that adopt an object owner's authority into a user space.

This API provides information similar to that provided by the Display Program Adopt (DSPPGMADP) command.

Authorities and Locks

User Space Authority
　　　*CHANGE

Authority to Library Containing User Space
　　　*EXECUTE

User Profile Authority
　　　*OBJMGT

Required Parameter Group

Qualified user space name
　　　INPUT; CHAR(20)

　　　The name of the existing user space to which the list of objects that adopt a user's authority is returned. The first 10 characters specify the user space name, and the second 10 characters specify the library.

　　　You can use these special values for the library name:

*CURLIB　　　　The current library is used to locate the user space. If there is no current library, QGPL (general purpose library) is used.

*LIBL　　　　　The library list is used to locate the user space.

Format name
　　　INPUT; CHAR(8)

　　　The name of the format that returns information on the objects that adopt a user's authority.

　　　You can specify these formats:

"OBJP0100 Format" on page 66　　　　Each entry contains the object name, library, type, object in use indicator, auxiliary storage pool (ASP) device name of library, and ASP device name of object.

"OBJP0110 Format" on page 67　　　　This format only returns path names for objects in directories. Each entry contains the offset to the path name, the length of the path name, ASP device name of object, and the path name value.

"OBJP0200 Format" on page 67　　　　Each entry contains the same information as format OBJP0100 plus the object attribute and descriptive text.

User profile name
　　　INPUT; CHAR(10)

　　　The user name for which the list of objects that adopt the user's authority is returned.

　　　You can specify the following special value:

*CURRENT　　　The list of objects that adopt the authority of the user currently running is returned. If *CURRENT is used, the name of the current user is returned in the list header section of the user space.

Object type
　　　INPUT; CHAR(10)

　　　The type of object for which the list of objects that adopt the user's authority is returned.

　　　You can specify only the following special values:

*ALL　　　　　Return entries for all object types that adopt authority that is supported by the requested format name.

*PGM　　　　　Return entries for programs that adopt authority.

*SQLPKG	Return entries for SQL packages that adopt authority.	
*SRVPGM	Return entries for service programs that adopt authority.	
*JVAPGM	Return entries for stream files that have attached JAVA programs that adopt authority.	

Continuation handle

INPUT; CHAR(20)

The handle used to continue from a previous call to this API that resulted in partially complete information. You can determine if a previous call resulted in partially complete information by checking the Information Status variable in the generic user space header following the API call.

If the API is not attempting to continue from a previous call, this parameter must be set to blanks. Otherwise, a valid continuation value must be supplied. The value may be obtained from the list header section of the user space used in the previous call. When continuing, the first entry in the returned list is the entry that immediately follows the last entry returned in the previous call.

Error code

I/O; CHAR(*)

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

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 67.

Input Parameter Section

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User space name specified
10	0A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	User name specified
38	26	CHAR(10)	Object type
48	30	CHAR(20)	Continuation handle

Header Section

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User name
10	0A	CHAR(20)	Continuation handle

OBJP0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Object name

Offset			
Dec	Hex	Type	Field
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Object in use
31	1F	CHAR(10)	ASP device name of library
41	29	CHAR(10)	ASP device name of object

OBJP0110 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

OBJP0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Object in use
31	1F	CHAR(10)	Attribute
41	29	CHAR(50)	Text description
91	5B	CHAR(10)	ASP device name of library
101	65	CHAR(10)	ASP device name of object

Field Descriptions

ASP device name of library. The auxiliary storage pool (ASP) device name where the object's library is stored. If the object's library is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

ASP device name of object. The auxiliary storage pool (ASP) device name where the object is stored. If the object is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

Attribute. The object attribute.

Continuation handle (header section). A continuation point for the API. This value is set based on the contents of the Information Status variable in the generic header for the user space. The following situations can occur:

- Information status-C. The information returned in the user space is valid and complete. No continuation is necessary and the continuation handle is set to blanks.
- Information status-P. The information returned in the user space is valid but incomplete. The user may call the API again, starting where the last call left off. The continuation handle contains a value which may be supplied as an input parameter in later calls.
- Information status-I. The information returned in the user space is not valid and incomplete. The content of the continuation handle is unpredictable.

Continuation handle (input section). Used to continue from a previous call to this API which resulted in partially complete information.

Format name. The name of the format used to return information on the objects that adopt authority.

Length of path name. The length, in bytes, of the path name.

Library name. The name of the library containing the user space or object.

Object name. The name of the object that adopts the user's authority.

Object in use. Whether the object is in use when the API tries to access it. If the object is in use, the API is not able to determine if the object adopts the user's authority. If the object is in use, this field is Y. If not, this field is N.

Object type.
- Input Section: The type of object for which the list of objects adopting the user's authority is returned.
- List Section: The type of object which adopts the user's authority.

Offset to path name. The offset in the user space to the start of the path name.

Path name. The path name of the object that adopts the user's authority.

The structure of the path name returned is:

Description	Type
CCSID of the returned path name	Binary(4)
Country or region ID	Char(2)
Language ID	Char(3)
Reserved field	Char(3)
Path type	Binary(4)
Number of bytes in the path name	Binary(4)
Path delimiter	Char(2)
Reserved field	Char(10)
Path name value	Char(*)

Text description. The text description of the object.

User name. The name of the owner of the object.

User name specified. The name of the user for which the list of objects that adopt the user's authority is returned.

User space library name specified. The name of the library that contains the user space.

User space name specified. The name of the user space to which the list of objects that adopt the users authority is returned.

Error Messages

Message ID	Error Message Text
CPF22FD E	Continuation handle not valid for API &1.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C31 E	Object type &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF811A E	User space &4 in &9 damaged.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

List Objects User Is Authorized to, Owns, or Is Primary Group of (QSYLOBJA) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	User profile name	Input	Char(10)
4	Object type	Input	Char(10)
5	Returned objects	Input	Char(10)
6	Continuation handle	Input	Char(20)
7	Error code	I/O	Char(*)

Optional Parameter Group:

8	Request list	Input	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The List Objects a User is Authorized to, Owns, or Is Primary Group of (QSYLOBJA) API puts a list of objects a user is authorized to, owns, or is the primary group owner for into a user space. The list of authorized objects only includes objects the user is specifically authorized to. The list does not include objects the user is authorized to because:

- The user is part of a group that is authorized
- The user can access the object using the public authority

- The object is secured with an authorization list the user is authorized to
- The user can access the object using adopted authority

This API provides information similar to that provided by the Display User Profile (DSPUSRPRF) command when specifying *OBJAUT, *OBJOWN, or *OBJPGP for the type parameter.

Authorities and Locks

User Space Authority
        *CHANGE

Authority to Library Containing User Space
        *EXECUTE

User Profile Authority
        *READ

Required Parameter Group

Qualified user space name
        INPUT; CHAR(20)

        The name of the existing user space used to return the list of objects a user is authorized to, owns, or is the primary group for. The first 10 characters specify the user space name, and the second 10 characters specify the library.

        You can use these special values for the library name:

CURLIB	The current library is used to locate the user space. If there is no current library, QGPL (general purpose library) is used.
LIBL	The library list is used to locate the user space.

Format name
        INPUT; CHAR(8)

        The name of the format used to list objects the owner is authorized to, owns, or is the primary group for.

        You can specify these formats:

"OBJA0100 Format" on page 72	Each entry contains the object name, library, type, authority holder indicator, ownership indicator, auxiliary storage pool (ASP) device name of library, and ASP device name of object.
"OBJA0110 Format" on page 73	This format only returns path names for objects in a directory. Each entry contains the offset to the path name, the length of the path name, type, authority holder indicator, ownership indicator, ASP device name of object, and the path name value.
"OBJA0200 Format" on page 73	Each entry contains the same information as format OBJA0100 plus the authority values.
"OBJA0210 Format" on page 74	This format only returns path names for objects in a directory. Each entry contains the same information as format OBJA0110 plus the authority values.
"OBJA0300 Format" on page 74	Each entry contains the same information as format OBJA0200 plus the object attribute and descriptive text.
"OBJA0310 Format" on page 75	This format only returns path names for objects in a directory. Each entry contains the same information as format OBJA0210 plus the attribute and descriptive text.

User profile name
        INPUT; CHAR(10)

The user name for which the list of objects is being returned.

You can specify the following special value:

CURRENT   The list of objects that the user currently running is authorized to, owns, or is the primary group for is returned. If *CURRENT is used, the name of the current user is returned in the list header section of the user space.

Object type
>   INPUT; CHAR(10)

>   The type of object the list of objects is returned for.

>   You can specify the following special value:

ALL   Return entries of all object types.

Returned objects
>   INPUT; CHAR(10)

>   The objects that are returned.

>   You can specify the following special values:

OBJAUT	The list of objects the user is authorized to is returned.
*OBJOWN	The list of objects the user owns is returned.
*BOTH	The list of objects the user is authorized to and owns is returned. The list of owned objects precedes the list of authorized objects.
*REQLIST	The values specified in the request list parameter is used.

Continuation handle
>   INPUT; CHAR(20)

>   The handle used to continue from a previous call to this API that resulted in partially complete information. You can determine if a previous call resulted in partially complete information by checking the Information Status variable in the generic user space header following the API call.

>   If the API is not attempting to continue from a previous call, this parameter must be set to blanks. Otherwise, a valid continuation value must be supplied. The value may be obtained from the list header section of the user space used in the previous call. When continuing, the first entry in the returned list is the entry that immediately follows the last entry returned in the previous call.

Error code
>   I/O; CHAR(*)

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

Optional Parameter Group

Request list
>   INPUT; CHAR(*)

>   The list of objects that are to be returned. This parameter can return more information than would be returned if the returned objects parameter was specified. This parameter is ignored unless the value in the returned objects parameter is *REQLIST.

>   You can specify the following values:
>   - *Number of values in the list.* - BINARY(4)

>   The number of values in the list of requests.

- List of requests - ARRAY(*) of CHAR(10)

  The values requested to return objects for a user.

  The possible values are:

  – *OBJAUT. - Returns the list of objects the user is authorized to.
  – *OBJOWN. - Returns the list of objects the user owns.
  – *OBJPGP. - Returns the list of objects the that the user is the primary group for.

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 76. When you retrieve list entry information for formats OBJA0100, OBJA0200, or OBJA0300 you must use the entry size returned in the generic header to access list entries as the size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. When you retrieve list entry information for formats OBJA0110, OBJA0210, OBJA0310 you must use the **offset to path name** of the current entry + the **length of path name** of the current entry to access the next entry.

Input Parameter Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name specified
10	0A	CHAR(10)	Library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	User profile name specified
38	26	CHAR(10)	Object type
48	30	CHAR(10)	Returned objects
58	3A	CHAR(20)	Continuation handle
78	4E	BINARY(4)	Offset to the request list
82	52	BINARY(4)	Number of values in the request list
86	56	CHAR(*)	List of requests

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User profile name
10	0A	CHAR(20)	Continuation handle
30	1E	BINARY(4)	Reason code

OBJA0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name

Offset			
Dec	Hex	Type	Field
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(1)	Ownership
32	20	CHAR(10)	ASP device name of library
42	2A	CHAR(10)	ASP device name of object

OBJA0110 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(1)	Ownership
20	14	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

OBJA0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(1)	Ownership
32	20	CHAR(10)	Authority value
42	2A	CHAR(1)	Authorization list management
43	2B	CHAR(1)	Object operational
44	2C	CHAR(1)	Object management
45	2D	CHAR(1)	Object existence
46	2E	CHAR(1)	Data read
47	2F	CHAR(1)	Data add
48	30	CHAR(1)	Data update
49	31	CHAR(1)	Data delete
50	32	CHAR(1)	Data execute
60	3C	CHAR(10)	Reserved
61	3D	CHAR(1)	Object alter
62	3E	CHAR(1)	Object reference

Offset			
Dec	**Hex**	**Type**	**Field**
63	3F	CHAR(10)	ASP device name of library
73	49	CHAR(10)	ASP device name of object

OBJA0210 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(1)	Ownership
20	14	CHAR(10)	Authority value
30	1E	CHAR(1)	Authorization list management
31	1F	CHAR(1)	Object operational
32	20	CHAR(1)	Object management
33	21	CHAR(1)	Object existence
34	22	CHAR(1)	Object alter
35	23	CHAR(1)	Object reference
36	24	CHAR(10)	Reserved
46	2E	CHAR(1)	Data read
47	2F	CHAR(1)	Data add
48	30	CHAR(1)	Data update
49	31	CHAR(1)	Data delete
50	32	CHAR(1)	Data execute
51	33	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

OBJA0300 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(1)	Ownership
32	20	CHAR(10)	Authority value
42	2A	CHAR(1)	Authorization list management
43	2B	CHAR(1)	Object operational

Offset		Type	Field
Dec	Hex		
44	2C	CHAR(1)	Object management
45	2D	CHAR(1)	Object existence
46	2E	CHAR(1)	Data read
47	2F	CHAR(1)	Data add
48	30	CHAR(1)	Data update
49	31	CHAR(1)	Data delete
50	32	CHAR(10)	Attribute
60	3C	CHAR(50)	Text description
110	6E	CHAR(1)	Data execute
111	78	CHAR(10)	Reserved
121	79	CHAR(1)	Object alter
122	7A	CHAR(1)	Object reference
123	7B	CHAR(10)	ASP device name of library
133	85	CHAR(10)	ASP device name of object

OBJA0310 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(1)	Ownership
20	14	CHAR(10)	Authority value
30	1E	CHAR(1)	Authorization list management
31	1F	CHAR(1)	Object operational
32	20	CHAR(1)	Object management
33	21	CHAR(1)	Object existence
34	22	CHAR(1)	Object alter
35	23	CHAR(1)	Object reference
36	24	CHAR(10)	Reserved
46	2E	CHAR(1)	Data read
47	2F	CHAR(1)	Data add
48	30	CHAR(1)	Data update
49	31	CHAR(1)	Data delete
50	32	CHAR(1)	Data execute
51	33	CHAR(10)	Reserved
61	3D	CHAR(10)	Attribute
71	47	CHAR(50)	Text description

Offset			
Dec	Hex	Type	Field
121	79	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

Field Descriptions

ASP device name of library. The auxiliary storage pool (ASP) device name where the object's library is stored. If the object's library is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

ASP device name of object. The auxiliary storage pool (ASP) device name where the object is stored. If the object is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

Attribute. The object's attribute.

Authority holder. Whether the object is an authority holder. If the object is an authority holder, this field is Y. If not, this field is N.

Authority value. The special value indicating the user's authority to the object.

This field contains one of the following values:

ALL	The user has all object (operational, management, existence, alter and reference) and data (read, add, update, delete, and execute) authorities to the object.
*CHANGE	The user has object operational and all data authorities to the object.
*USE	The user has object operational and data read and execute authorities to the object.
*EXCLUDE	The user has none of the object or data authorities to the object, or authorization list management authority.
USER DEF	The user has some combination of object and data authorities that do not relate to a special value. The individual authorities for the user should be checked to determine what authority the user has to the object. This value is returned if the user owns an object and all authority for the user to the object has been removed. If this happens, all individual authority fields are set to N.

Authorization list management. Whether the user has authorization list management authority to the object. If the user has the authority, this field is Y. If not, this field is N. This field is only valid if the object type is *AUTL.

Continuation handle (header section). A continuation point for the API. This value is set based on the contents of the Information Status variable in the generic header for the user space.

The following situations can occur:

- Information status-C. The information returned in the user space is valid and complete. No continuation is necessary and the continuation handle is set to blanks.
- Information status-P. The information returned in the user space is valid but incomplete. The user may call the API again, picking up where the last call ended. The continuation handle contains a value, that may be supplied as an input parameter in later calls.
- Information status-I. The information returned in the user space is not valid or complete. The contents of the continuation handle are unpredictable.

Continuation handle (input section). The handle used to continue from a previous call to this API that resulted in partially complete information.

Data add. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data delete. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data execute. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data read. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data update. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Format name. The name of the format used to list objects the user is authorized to or owns.

Length of path name. The length, in bytes, of the path name.

Library name. The name of the library containing the user space or object.

Library name specified. The name of the library that will contain the user space or object.

List of requests. The list of values requested in the list of requests parameter.

Number of values in the request list. The number of values that were specified in the list of requests.

Object alter. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object existence. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object management. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object name. The name of the object the user is authorized to, owns, or is the primary group for.

Object operational. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object reference. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object type.

Input Section	The type of object for which the list of authorized, owned, or primary group objects is returned.
List Section	The type of object the user is authorized to, owns, or is the primary group of.

Offset to path name. The offset in the user space to the start of the path name.

Offset to the request list. The offset to the specified list of requests.

Ownership. Whether the user owns the object or is the primary group for the object. If the user owns the object, this field is Y. If the user is the primary group for the object, this field is G. Otherwise, this field is N.

Path name. The path name of the object the user owns, is authorized to, or is the primary group for.

The structure of the path name returned is:

Description	Type
CCSID of the returned path name	Binary(4)
Country or region ID	Char(2)
Language ID	Char(3)
Reserved field	Char(3)
Flag byte	Binary(4)
Number of bytes in the path name	Binary(4)
Path delimiter	Char(2)
Reserved field	Char(10)
Path name value	Char(*)

Primary group. The name of the user who is the primary group for the authorization list or object. If there is no primary group for the authorization list or object, this field will contain a value of *NONE.

Reason code. The reason code describing why the returned list is only a subset. The following values can be returned:

- Reason code 0000. The list returned in the user space contains all objects meeting the search criteria.
- Reason code 0001. Objects were found that meet the search criteria but could not be included in the returned list. The requested format could not handle path names for directory objects.
- Reason code 0002. Objects were found that meet the search criteria but could not be included in the returned list. The requested format could not handle objects found in library QSYS.
- Reason code 0003. Directory objects were found but did not have links to them.

Reserved. An ignored field.

Returned objects. The objects that are returned.

Text description. The text description of the object.

User profile name. The user name used to return the list of objects.

User profile name specified. The user name for which the list of objects is returned.

User space name. The name of the user space used to return the list of objects.

User space name specified. The name of the user space in which the list of objects is returned.

Error Messages

Message ID	Error Message Text
CPF22FC E	Value &1 not valid when specifying objects to be returned by API &2.
CPF22FD E	Continuation handle not valid for API &1.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
CPF222A E	Value &1 not valid when specifying a list of requests for API &2.

Message ID	Error Message Text
CPF222B E	The requested list parameter is not specified for API &1.
CPF222C E	&1 is not valid for the number of requested list values for API &2.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C31 E	Object type &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

List Users Authorized to Object (QSYLUSRA) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Qualified object name	Input	Char(20)
4	Object type	Input	Char(10)
5	Error code	I/O	Char(*)

Optional Parameter Group:

6	ASP device	Input	Char(10)

Default Public Authority: *USE
Threadsafe: Yes

The List Users Authorized to Object (QSYLUSRA) API puts a list of users privately authorized to an object, including an authorization list, into a user space. The information returned is the authority as it exists for the object. Any authority the process has to the object through its group or adopted authority is not included. *PUBLIC authority to the object is also returned in the first list entry of the user space.

If the object is a database file, an indication of whether the file has field authorities is returned.

This API provides information similar to that provided by the Display Authorization List (DSPAUTL) command or the Display Object Authority (DSPOBJAUT) command.

Authorities and Locks

User Space Authority
    *CHANGE

Authority to Library Containing User Space
    *EXECUTE

Specified Object or Authorization List Authority
    *OBJMGT

Required Parameter Group

Qualified user space name
        INPUT; CHAR(20)

        The name of the existing user space used to return the list of authorized users to the object. The first 10 characters specify the user space name, and the second 10 characters specify the library.

        You can use these special values for the library name:

CURLIB   The current library is used to locate the user space. If there is no current library, QGPL (general purpose library) is used.
LIBL   The library list is used to locate the user space.

Format name
        INPUT; CHAR(8)

        The name of the format used to list authorized users.

        You can specify this format:

"USRA0100 Format" on page 81   Each entry contains the user name and authority values.

Qualified object name
        INPUT; CHAR(20)

        The name of the object for which the list of authorized users is returned. The first 10 characters specify the object name, and the second 10 characters specify the library.

        You can use these special values for the library name:

CURLIB   The current library is used to locate the object. If there is no current library, QGPL (general purpose library) is used.
LIBL   The library list is used to locate the object.

Object type
        INPUT; CHAR(10)

        The type of object for which the list of authorized users is returned.

Error code
        I/O; CHAR(*)

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

Optional Parameter Group

ASP device
        INPUT; CHAR(10)

        The name of the auxiliary storage pool (ASP) device in which to search for the library that contains the object.

        The valid values are:

*   All ASPs associated with the job will be searched. This is the default value if the parameter is not specified.
SYSBAS   The system ASP and all basic user ASPs will be searched.

ALL All ASPs that are currently available will be searched.

ASP device name The specified ASP will be searched.

If *CURLIB or *LIBL is specified for the library then the ASP device parameter must be specified as *.

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 82.

Input Parameter Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name specified
10	0A	CHAR(10)	Library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	Object name
38	26	CHAR(10)	Library name specified
48	30	CHAR(10)	Object type
58	3A	CHAR(10)	ASP device

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name specified
20	14	CHAR(10)	Object type
30	1E	CHAR(10)	Owner name
40	28	CHAR(10)	Authorization list
50	32	CHAR(10)	Primary group
60	3C	CHAR(1)	Field authorities
61	3D	CHAR(10)	ASP device name of library
71	47	CHAR(10)	ASP device name of object

USRA0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User profile name
10	0A	CHAR(10)	Authority value
20	14	CHAR(1)	Authorization list management
21	15	CHAR(1)	Object operational

Offset			
Dec	Hex	Type	Field
22	16	CHAR(1)	Object management
23	17	CHAR(1)	Object existence
24	18	CHAR(1)	Data read
25	19	CHAR(1)	Data add
26	1A	CHAR(1)	Data update
27	1B	CHAR(1)	Data delete
28	1C	CHAR(1)	Data execute
29	1D	CHAR(10)	Reserved
39	27	CHAR(1)	Object alter
40	28	CHAR(1)	Object reference

Field Descriptions

ASP device name of library. The auxiliary storage pool (ASP) device name where the object's library is stored. If the object's library is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

ASP device name of object. The auxiliary storage pool (ASP) device name where the object is stored. If the object is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

Authority value. The user's authority to the object.

This field contains one of the following values:

*ALL	The user has all object (operational, management, existence, alter, and reference) and data (read, add, update, delete, and execute) authorities to the object.
*CHANGE	The user has object operational and all data authorities to the object.
*USE	The user has object operational and data read and execute authorities to the object.
*EXCLUDE	The user has none of the object or data authorities to the object, or authorization list management authority to the authorization list.
*AUTL	The public authority for the object comes from the public authority on the authorization list securing the object. This value can only be returned if there is an authorization list securing the object and the authorized user is *PUBLIC.
USER DEF	The user has some combination of object and data authorities that do not relate to a special value. The individual authorities for the user should be checked to determine what authority the user has to the object.

Authorization list. The name of the authorization list securing the object. If there is no authorization list securing the object, this field is *NONE.

Authorization list management. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N. This field is only valid if the object type is *AUTL.

Data add. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data delete. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data execute. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data read. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data update. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Field authorities. Whether the object has field authorities. If the object is a database file and it has field authorities, this field is Y. If not, this field is N. This field is only valid if the object type is *FILE. To see the field authorities for a database file, do DSPOBJAUT OBJ(your_lib/your_dbfile) OBJTYPE(*FILE) AUTTYPE(*FIELD).

Format name. The name of the format used to list users authorized to the object.

Library name specified. The name of the library the object containing the authorization list is in.

Primary group. The name of the user that is the primary group for the object. If there is not a primary group for the object, the field will contain *NONE.

Object alter. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object existence. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object management. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object name. The name of the object for which the list of authorized users is returned.

Object operational. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object reference. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object type. The type of object for which the list of authorized users is returned.

Owner. The name of the owner of the object. If all authority for the owner is removed, no list entry is returned for the owner.

Reserved. An ignored field set to hexadecimal zeros.

User profile name. The name of the user authorized to the object.

This field can contain the following special value:

*PUBLIC   Public authority (authority used by users not privately authorized) to the object. This is the first entry in the list data section.

User space name specified. The name of the user space used to return the list of users authorized to the object.

Error Messages

Message ID	Error Message Text
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C31 E	Object type &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF811A E	User space &4 in &9 damaged.
CPF980B E	Object &1 in library &2 not available.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9814 E	Device &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9825 E	Not authorized to device &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9873 E	ASP status is preventing access to object.

API introduced: V4R2

Open List of Authorized Users (QGYOLAUS) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	List information	Output	Char(80)
4	Number of records to return	Input	Binary(4)
5	Format name	Input	Char(8)
6	Selection criteria	Input	Char(10)
7	Group profile name	Input	Char(10)
8	Error Code	I/O	Char(*)

Optional Parameter:

9	Profile name	Input	Char(10)

Default Public Authority: *USE
Threadsafe: No

The Open List of Authorized Users (QGYOLAUS) API provides information about the authorized users of the system. It returns a list of authorized user names that meet the selection criteria specified by the caller of the API and information about those users. This API provides information similar to the Display Authorized Users (DSPAUTUSR) command and the Retrieve Authorized Users (QSYRAUTU) API.

Differences between QSYRAUTU and QGYOLAUS

The QGYOLAUS API returns the same information that the Retrieve Authorized Users (QSYRAUTU) API provides, but takes a complete snapshot at once and allows subsequent records to be obtained through the Get List Entries (QGYGTLE) API.

Authorities and Locks

Authority to User Profiles in List of Authorized Users
    *READ

> **Note:** Only those profiles to which you have *READ authority are returned in the list.

Required Parameter Group

Receiver variable
    OUTPUT; CHAR(*)

    The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
    INPUT; BINARY(4)

    The length of the receiver variable. If the length is larger than the size of the receiver variable, the results are not predictable.

List Information
    OUTPUT; CHAR(80)

    Information about the list created by this program. For a description of the layout of this parameter, see Format of Open List Information.

Number of records to return
    INPUT; BINARY(4)

    The number of records in the list to put into the receiver variable after filtering and sorting has been done.

    If -1 is specified for this parameter, the entire list is built synchronously.

    If 0 is specified for this parameter, the entire list is built asynchronously in a server job.

    If a positive number of records to return is specified, at least that many records will be built synchronously and the remainder will be built asynchronously in a server job.

Format name
    INPUT; CHAR(8)

    The name of the format that is used to return information about the authorized users.

    You can specify these formats:

"AUTU0100 Format" on page 87	Each entry contains the user name, an indicator that specifies whether the user is a user profile or a group profile, and an indicator that specifies whether the user is a group that has members.
"AUTU0150 Format" on page 87	Each entry contains the same information as AUTU0100 plus the text description for the user.
"AUTU0200 Format" on page 87	Each entry contains the same information as AUTU0100 plus group profiles are returned for users who are members of one or more groups.
"AUTU0250 Format" on page 87	Each entry contains the same information as AUTU0200 plus the text description for the user.

Selection criteria
    INPUT; CHAR(10)

   This parameter specifies which users are returned. Possible special values follow:

*ALL	All user profile names and group profile names are returned. This is the same list of users that are returned by the List Authorized Users (QSYLAUTU) API.
*USER	User names that are not group profiles are returned. (Users that do not have a GID specified in their user profiles.)
*GROUP	User names that are group profiles are returned. (Users that have a GID specified in their user profiles.)
*MEMBER	User names that are members of the group specified by the group profile name parameter are returned. The users who do not have any group profiles can be retrieved by specifying *NOGROUP for the group profile name. The group profile name parameter must contain a valid group profile name or *NOGROUP when the selection criteria is *MEMBER.

Group profile name
    INPUT; CHAR(10)

   The group profile whose members are to be returned. The profile specified must exist and must be a group profile.

   A group profile name or *NOGROUP is required if *MEMBER is specified for the selection criteria. The group profile name must be *NONE if the selection criteria is not *MEMBER.

*NONE	No group profile is specified.
group name	Users who are a member of this group are returned.
*NOGROUP	Users who are not a member of any group are returned.

Error code
    I/O; CHAR(*)

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

Optional Parameter

Profile name
    INPUT; CHAR(10)

   The profile names to include in the list. The selection criteria determines which users are included in the list. Specifying a profile name can further limit the names that are returned.

   The profile name can be a simple name, a generic name, or the special value *ALL. If not provided, *ALL is used as a default. Possible values follow:

*ALL	All profiles are listed.
Profile name	If a generic profile name is specified, the profiles that match the generic name are returned. If a simple profile name is specified, only that profile is returned.

Receiver Variable Description

The following tables describe the order and format of the data returned in the receiver variable for each profile name in the list. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 88.

AUTU0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator

AUTU0150 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator
12	0C	CHAR(50)	Text description

AUTU0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator
12	0C	BINARY(4)	Number of group profiles
16	10	ARRAY(16) of CHAR(10)	Group profiles

AUTU0250 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator
12	0C	CHAR(50)	Text description
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	Number of group profiles
68	44	ARRAY(16) of CHAR(10)	Group profiles

Field Descriptions

Group members indicator. Whether this user is a group that has members. Possible values follow:

0	The user is not a group, or is a group but does not have any members.
1	The user is a group that has members.

Group profiles. The array of group profiles for the user. The number of group profiles field indicates how many entries are in the array.

Number of group profiles. The number of group profiles that are returned in the group profiles field. The number of group profiles will be zero if the user is not a member of any group.

Profile name. The name of an authorized user for whom information is returned.

Reserved. An ignored field.

Text description. The descriptive text for the user profile.

User or group indicator. Whether this user is a user profile or a group profile. Possible values follow:

0	User profile
1	Group profile

Error Messages

Message ID	Error Message Text
CPF22B4 E	Group profile &1 not found.
CPF22B7 E	Profile &1 is not a group profile.
CPF22E0 E	Group profile name cannot be *NONE when selection criteria is *MEMBER.
CPF22ED E	Group profile name must be *NONE when selection criteria is not *MEMBER.
CPF22EE E	Selection criteria is not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0027 E	&1 is not valid for number of records to return.

API introduced: V4R1

Release Profile Handle (QSYRLSPH, QsyReleaseProfileHandle) API

»

Required Parameter Group for QSYRLSPH:

1	Profile handle	Input	Char(12)

Optional Parameter for QSYRLSPH:

2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

Syntax for QsyReleaseProfileHandle:

```
#include <qsyphandle.h>

void QsyReleaseProfileHandle
        (unsigned char * Profile_handle,
         void          * Error_code);
```

Service Program: QSYPHANDLE
Default Public Authority: *USE
Threadsafe: Yes

《

The Release Profile Handle (OPM, QSYRLSPH; ILE, QsyReleaseProfileHandle) API validates a given profile handle and then releases it. To use the user profile represented by the deleted profile handle in the job again, you must call the Get Profile Handle (QSYGETPH, QsyGetProfileHandle) API to generate a new profile handle for the user profile.

Your application must perform any needed cleanup work like closing files and deallocating objects. The Release Profile Handle API only releases the profile handle; it does not perform any cleanup in the job.

Authorities and Locks

None

Required Parameter

Profile handle
        INPUT; CHAR(12)

        The profile handle to be released.

Optional Parameter

Error code
        I/O; CHAR(*)

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

        This parameter is optional for QSYRLSPH API and is omissable for the QsyReleaseProfileHandle API.

Usage Notes

If the profile handle to be released is not valid (for example, the profile handle was not generated in this job or had previously been released), the call to this API will not fail. Message CPF22E7 (ProfileHandle is not valid) will be in the joblog as a diagnostic message.

Error Messages

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Remove All Profile Tokens (QsyRemoveAllPrfTkns) API

≫

Syntax for QsyRemoveAllPrfTkns:

```
#include <qsyptkn.h>

void QsyRemoveAllPrfTkns
      (void            *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Remove All Profile Tokens (QsyRemoveAllPrfTkns) API provides an interface to remove all profiles on the system. This may be useful if the maximum number of profile tokens have been generated for the system (message CPF4AAA was sent or a PS-M security audit entry was sent). The most likely reason for this to happen is that someone is attempting to lock up parts of the system by generating multitudes of profile tokens. This API provides an alternative to restarting the system. After calling this API, the administrator may want to analyze the audit log to determine who is attempting to lock up the system.

Authorities and Locks

API Public Authority
   *USE

Special authority required
   *ALLOBJ and *SECADM

Required Parameter

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.
CPF2225 E	Not able to allocate internal system object.
CPF3CF1 E	Error code parameter not valid.

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R4

Remove All Profile Tokens For User (QsyRemoveAllPrfTknsForUser) API

≫

Syntax for QsyRemoveAllPrfTknsForUser:

```
#include <qsyptkn.h>

void QsyRemoveAllPrfTknsForUser
      (char            *User_profile,
       void            *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Remove All Profile Tokens For User (QsyRemoveAllPrfTknsForUser) API provides an interface to remove all profile tokens that have been generated for a specific user profile. You may want to remove all profile tokens for a user profile if security information has changed for the user profile (for example, the password or group list).

Authorities and Locks

API Public Authority
> *USE

Special authority required
> *SECADM

User profile authority
> *OBJMGT and *USE

Required Parameter Group

User profile
> INPUT; CHAR(10)

> The name of the user profile for which to remove profile tokens.

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.
CPF222E E	&1 special authority is required.
CPF2217 E	Not authorized to user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF3CF1 E	Error code parameter not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Remove Profile Token (QsyRemovePrfTkn) API

≫

Syntax for QsyRemovePrfTkn:

```
#include <qsyptkn.h>

void QsyRemovePrfTkn
      (unsigned char  *Profile_token,
       void           *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Remove Profile Token (QsyRemovePrfTkn) API removes the specified profile token. The profile token will no longer be valid for use with other profile token APIs.

Authorities and Locks

API Public Authority
    *USE

Required Parameter Group

Profile token
    INPUT; CHAR(32)

    The profile token to be removed.

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.

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF3CF1 E	Error code parameter not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R4

Remove Profile Tokens (QSYRMVPT) API

Required Parameter Group:

1	Remove option	Input	Char(10)
2	Error code	I/O	Char(*)

Optional Parameter:

3	Profile token	Input	Char(32)

Default Public Authority: *USE
Threadsafe: Yes

The Remove Profile Tokens (QSYRMVPT) API removes all profile tokens, removes all profile tokens for a specific user profile, or removes a specific profile token. When a profile token is removed, it is no longer valid for use with other profile token APIs.

This API can be used to remove all profile tokens on the system. This may be useful if the maximum number of profile tokens have been generated for the system (message CPF4AAA was sent or a PS-M security audit entry was sent). The most likely reason for this to happen is that someone is attempting to lock up parts of the system by generating multitudes of profile tokens. This API provides an alternative to restarting the system. After calling this API, the administrator may want to analyze the audit log to determine who is attempting to lock up the system.

This API can be used to remove all profile tokens that have been generated for a user profile. You may want to remove all profile tokens for a user profile if security information has changed for the user profile (for example, the password or group list).

Authorities and Locks

*Authority, if *ALL is specified*
      *ALLOBJ and *SECADM

Authority, if specific user is specified
      *SECADM

User profile authority, if specific user is specified
      *OBJMGT and *USE

Required Parameter Group

Remove option
      INPUT; CHAR(10)

      Whether all profile tokens are being removed, all profile tokens for a given user are being removed, or a specific profile token is being removed.

One of the following values may be specified:

ALL All profile tokens will be removed.

PRFTKN The specified profile token will be removed. If this value is specified, the optional parameter that contains the profile token must be specified.

User name The name of the user profile for which to remove all profile tokens.

Error code
>   I/O; CHAR(*)

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

Optional Parameter Group

Profile token
>   INPUT; CHAR(32)

The profile token to be removed. This parameter is required only if *PRFTKN is specified in the Remove option parameter. If the Remove option is not *PRFTKN and this parameter is specified, then this parameter must contain *NONE.

Error Messages

Message ID	Error Message Text
CPF2204 E	User profile &1 not found.
CPF222E E	&1 special authority is required.
CPF2217 E	Not authorized to user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF2274 E	Profile token is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R4

Remove User Application Information (QsyRemoveUserApplicationInfo) API

≫

Syntax for QsyRemoveUserApplicationInfo:

```
#include <qsyusrin.h>

void QsyRemoveUserApplicationInfo
      (char          *User_profile,
       char          *Application_information_ID,
       int           *Length_of_application_information_ID,
       void          *Error_code);
```

Service Program: QSYUSRIN
Default Public Authority: *USE
Threadsafe: No

《

The Remove User Application Information (QsyRemoveUserApplicationInfo) API removes the specified application information from the specified user profile.

The Change User Profile exit programs are not called from this API.

Authorities and Locks

If the user profile parameter is not *CURRENT or the user profile currently running, then the user profile that calls this API must have *SECADM special authority and *OBJMGT and *USE authorities to the user profile.

Required Parameter Group

User profile
    INPUT; CHAR(10)

    The user profile for which the application information will be removed. The special value *CURRENT may be specified to remove application information for the user profile that calls this API.

Application information ID
    INPUT; CHAR(*)

    The ID for the application information to be removed. The following can be specified for the application information ID:

    generic*
        All application information IDs that have IDs beginning with the generic string will be removed.

    application information ID
        Specific application information ID will be removed.

Length of application information ID
    INPUT; BINARY(4)

    The length of the application information ID that is specified in the application information ID parameter. The length of the application information ID must be a value from 1 to 200.

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.
CPF2213 E	Not able to allocate user profile &1.
CPF226C E	Not authorized to perform function.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF4AA2 E	Application information ID &1 does not exist.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

Reset Profile Attributes (QSYRESPA) API

Required Parameter:

1	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Reset Profile Attributes (QSYRESPA) API resets four attributes of system-supplied user profiles. Only system-supplied user profiles that cannot be changed using the Change User Profile (CHGUSRPRF) command are modified by the QSYRESPA API.

The following user profile attributes are reset:
- The password value is set to *NONE.
- The status value is set to *ENABLED.
- The password expired value is set to *NO
- The special authorities will be set to the appropriate values for the each system supplied profile.

The following system supplied user profiles have their attributes reset:

```
QAUTPROF    QCLUMGT     QCLUSTER    QCOLSRV
QDBSHR      QDBSHRDO    QDFTOWN     QDIRSRV
QDLFM       QDOC        QDSNX       QFNC
QGATE       QIPP        QLPAUTO     QLPINSTALL
QMGTC       QMSF        QNTP        QPEX
QPPM400     QSNADS      QSPL        QSPLJOB
QSRVAGT     QTCP        QTFTP       QTSTRQS
QYCMCIMOM   QYPSJSVR
```

If errors are encountered processing an individual user profile, diagnostic message CPD22BD is issued and processing continues with the next user profile. If any profiles cannot not be processed, escape message CPF22F0 is sent to notify the caller to look at the diagnostic messages.

Authorities and Locks

The caller of the API must have *SECADM and *ALLOBJ special authorities.

Required Parameter

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.
CPF2225 E	Not able to allocate internal system object.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3C90 E	Literal value cannot be changed.

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

API introduced: V4R5

Retrieve Authorized Users (QSYRAUTU) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Returned records feedback information	Output	Char(16)
4	Format name	Input	Char(8)
5	Selection criteria	Input	Char(10)
6	Starting profile name	Input	Char(10)
7	Starting profile option	Input	Char(1)
8	Group profile name	Input	Char(10)
9	Error Code	I/O	Char(*)

Optional Parameter:

10	Ending profile name	Input	Char(10)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Authorized Users (QSYRAUTU) API provides information about the authorized users of the system. It returns a list of authorized user names that meet the selection criteria specified by the caller of the API and information about those users. This API provides information similar to the Display Authorized Users (DSPAUTUSR) command.

Authorities and Locks

Authority to User Profiles in List of Authorized Users
 *READ

 Note: Only those profiles to which you have *READ authority are returned in the list.

Required Parameter Group

Receiver variable
 OUTPUT; CHAR(*)

 The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
 INPUT; BINARY(4)

 The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable.

Returned records feedback information
 OUTPUT; CHAR(16)

Information about the entries that are returned in the receiver variable.

See "Format of Returned Records Feedback Information" on page 100 for details.

Format name
> INPUT; CHAR(8)

> The name of the format that is used to return information about the authorized users.

> You can specify these formats:

"AUTU0100 Format" on page 99	Each entry contains the user name, an indicator specifying whether the user is a user profile or a group profile and an indicator specifying whether the user is a group that has members.
"AUTU0150 Format" on page 100	Each entry contains the same information as AUTU0100 plus the text description for the user.
"AUTU0200 Format" on page 100	Each entry contains the same information as AUTU0100 plus group profiles are returned for users who are members of one or more groups.
"AUTU0250 Format" on page 100	Each entry contains the same information as AUTU0200 plus the text description for the user.

Selection criteria
> INPUT; CHAR(10)

> The users that are returned.

> Possible values follow:

*ALL	All user profile and group profile names are returned. This is the same list of users that is returned by the List Authorized Users (QSYLAUTU) API.
*USER	User names that are not group profiles are returned. (Users that do not have a GID specified in their user profile.)
*GROUP	User names that are group profiles are returned. (Users that have a GID specified in their user profile.)
*MEMBER	User names that are members of the group specified by the group profile name parameter are returned. The users who do not have any group profiles can be retrieved by specifying *NOGROUP for the group profile name. Only user names that are not group profiles are returned. The group profile name parameter must contain a valid group profile name or *NOGROUP when the selection criteria parameter is *MEMBER.

Starting profile name
> INPUT; CHAR(10)

> The profile name at which to start the listing. The profile names are listed alphabetically.

> Possible values follow:

*FIRST	Profiles are returned starting with the first profile alphabetically.
profile name	If an exact match for the starting profile name is found, the starting profile option parameter indicates whether that profile name is returned. If an exact match for the starting profile name is not found, the listing begins with the first existing profile name after the specified starting profile name. For example, assume the authorized users are ED, FRANK, and MARY. If F is specified for the starting user profile, the list returned would be FRANK and MARY.

Starting profile option
> INPUT; CHAR(1)

> This parameter indicates whether the starting profile name is returned when an exact match for the starting profile name is found. Possible values follow:

| 0 | Profile names greater than the starting profile are returned. |
| 1 | Profile names equal to and greater than the starting profile name are returned. |

Group profile name
> INPUT; CHAR(10)

> The group profile whose members are to be returned. The profile that is specified must exist and must be a group profile.

> A group profile name or *NOGROUP is required if *MEMBER is specified for the selection criteria parameter. The group profile name must be *NONE if the selection criteria parameter is not *MEMBER.

*NONE	No group profile is specified.
group name	Users who are a member of this group are returned.
*NOGROUP	Users who are not a member of any group are returned.

Error code
> I/O; CHAR(*)

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

Optional Parameter

Ending profile name
> INPUT; CHAR(10)

> The profile name at which to end the listing. Specifying an ending profile name can limit the names that are returned.

> If this parameter is not provided, *LAST is used as a default. Possible values are:

| *LAST | Profiles up to and including the last profile are returned. |
| *Profile name* | The last profile name to be included in the list. |

Receiver Variable Description

The following tables describe the order and format of the data returned in the receiver variable for each profile name in the list. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 101.

AUTU0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator

AUTU0150 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator
12	0C	CHAR(50)	Text description

AUTU0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator
12	0C	BINARY(4)	Number of group profiles
16	10	ARRAY(16) of CHAR(10)	Group profiles

AUTU0250 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Profile name
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(1)	Group members indicator
12	0C	CHAR(50)	Text description
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	Number of group profiles
68	44	ARRAY(16) of CHAR(10)	Group profiles

Format of Returned Records Feedback Information

For a description of the fields in this format, see "Field Descriptions" on page 101.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of profile names
12	C	BINARY(4)	Entry length for each profile returned

Field Descriptions

Bytes available.

The number of bytes of data available to be returned to the user in the receiver variable. If all data is returned, bytes available is the same as the number of bytes returned. If the receiver variable was not big enough to contain all of the data, this value is estimated based on the total number of authorized users of the system and the format specified.

Bytes returned. The number of bytes of data returned to the user in the receiver variable. This is the lesser of the number of bytes available to be returned or the length of the receiver variable.

Entry length for each profile returned. The entry length, in bytes, of each element in the list of profile names. A value of zero is returned if the list is empty.

Group members indicator. Whether this user is a group that has members. Possible values follow:

0   The user is not a group, or is a group but does not have any members.
1   The user is a group that has members.

Group profiles. The array of group profiles for the user. The number of group profiles field indicates how many entries are in the array.

Number of group profiles. The number of group profiles returned in the group profiles field. The number of group profiles will be zero if the user is not a member of any groups.

Number of profile names. The number of complete entries in the list of profile names. A value of zero is returned if the list is empty.

Profile name. The name of an authorized user for whom information is returned.

Reserved. An ignored field.

Text description. The descriptive text for the user profile.

User or group indicator. Whether this user is a user profile or a group profile. Possible values follow:

0   User profile
1   Group profile

Error Messages

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF22B4 E	Group profile &1 not found.
CPF22B7 E	Profile &1 is not a group profile.
CPF22E0 E	Group profile name cannot be *NONE when selection criteria is *MEMBER.
CPF22ED E	Group profile name must be *NONE when selection criteria is not *MEMBER.
CPF22EE E	Selection criteria is not valid.
CPF22EF E	Starting profile option must be 0 or 1.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R7

Retrieve Encrypted User Password (QSYRUPWD) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format	Input	Char(8)
4	User profile name	Input	Char(10)
5	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: No

The Retrieve Encrypted User Password (QSYRUPWD) API returns to the caller the encrypted password data for the specified user profile. This API works with the Set Encrypted User Password (QSYSUPWD) API in that the APIs allow the user to more easily mirror the user profile activity on a second system based on the activity at the first system.

The data returned by the QSYRUPWD APIs should not be sent to a system that is at a different release or at a different password level. If data from this API is applied to a down-level system or a system with a different password level, unexpected changes to the user's password data could occur. For example, if the encrypted password data is retrieved from a system operating at password level 3 and is set on a system operating at password level 0 (or a pre-V5R1 system), the user profile's password is changed to *NONE. No checks are made to enforce these recommendations.

If the local password management (LCLPWDMGT) value for the specified user profile is *NO, then the local i5/OS password will be set to *NONE when the QSYSUPWD API is called. Also, if the LCLPWDMGT value is *NO for the user profile on the system where the QSYSUPWD API is called, then the local i5/OS password will be set to *NONE.

Except for the iSeries Support for Windows Network Neighborhood (iSeries NetServer) password, the QSYRUPWD API does not retrieve product-level encrypted data that may be associated with a user profile.

Note: If an error occurs while attempting to retrieve the iSeries NetServer password, the CPF22F0 error will be returned and no encrypted password data is returned.

Authorities and Locks

User Profile Authority
　　　　*ALLOBJ and *SECADM

API Public Authority
　　　　*EXCLUDE

Required Parameter Group

Receiver variable
        OUTPUT; CHAR(*)

The variable used to return the information about the user. The necessary size of this receiver variable can be obtained by calling the QSYRUPWD API with the length of receiver variable set to 8 bytes. The bytes available value that is returned in this receiver variable will indicate the necessary size of the receiver variable. The receiver variable format is defined in "UPWD0100 Format."

Length of receiver variable
        INPUT; BINARY(4)

The length of the receiver variable. This value must be at least 8 bytes in length. To obtain all information necessary to call the QSYSUPWD API, you must use a receiver variable at least as long as the bytes available value that is returned by this API.

Format
        INPUT; CHAR(8)

The name of the format that is used to return the user's encrypted password.

The following value is allowed:

"UPWD0100 Format"   Encrypted password is returned.

User profile name
        INPUT; CHAR(10)

The name of the user for whom the encrypted password will be returned.

Error code
        I/O; CHAR(*)

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

Format of Receiver Variable

The following tables describe the receiver variable that is returned by the QSYRUPWD API. This receiver variable is used as input to the QSYSUPWD API (first parameter). The receiver variable cannot be changed in any way prior to passing the data to the QSYSUPWD API. If this data is changed, the QSYSUPWD API will not be able to successfully change the password for the user.

For detailed descriptions of the fields in this table, see "Field Descriptions" on page 104.

UPWD0100 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User profile name
18	12	CHAR(*)	Encrypted user password data

Field Descriptions

Bytes available. The number of bytes of data available to be returned to the user. Bytes available may increase from release to release but will always be a minimum of 2000 bytes. This field should be used to set the length of receiver variable input parameter. If the bytes available field is greater than the bytes returned field, the receiver variable cannot successfully be used as input to the QSYSUPWD API as not all encrypted password data will be returned by this API.

Bytes returned. The number of bytes of data returned to the user in the receiver variable.

Encrypted user password data. The encrypted password data for the specified user profile.

User profile name. The name of the user profile for which information is being returned.

Error Messages

Message ID	Error Message Text
CPF2203 E	User profile &1 not correct.
CPF2225 E	Not able to allocate internal system object.
CPF222E E	&1 special authority is required.
CPF22F0 E	Unexpected errors occurred during processing.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9801 E	Object &2 in library &3 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R7

Retrieve Object Signatures (QYDORTVO, QydoRetrieveDigitalSignatures)API

Required Parameter Group:

1	Object path name	Input	Char(*)
2	Length of object path name	Input	Binary(4)
3	Format of object path name	Input	Char(8)
4	Receiver	Output	Char(*)
5	Length of receiver variable	Input	Binary(4)
6	Format of receiver variable	Input	Char(8)
7	Error code	I/O	Char(*)

Service Program Name: QYDORTV1
Default Public Authority: *USE
Threadsafe: No

The Retrieve Object Signatures (OPM, QYDORTVO; ILE, QydoRetrieveDigitalSignatures) API retrieves certificate information from a signed iSeries object.

Authorities and Locks

Authority Required

For objects in a library:

- *READ authority to the object
- *OBJOPR and *EXECUTE authority to the library.

For objects in a directory:

- *R authority to the object
- *X authority to each directory in the path.

Locks   Object will be locked shared allow read.

Required Parameter Group

Object path name
     INPUT; CHAR(*)

The name of the object from which you want to retrieve signatures. If the object is not in a library, the name may be relative to the current directory or may specify the entire path name. If the object is in a library, the name must be in the form '/QSYS.LIB/libname.LIB/objname.objtype' if you are using format OBJN0100 object path naming. For example, to sign a program named NEWEMPL in library PAYROLL, the qualified object name would be '/QSYS.LIB/PAYROLL.LIB/NEWEMPL.PGM'. Also, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job if you are using format OBJN0100 object path naming. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

If the object is in the QSYS file system, the object type must be *PGM, *SRVPGM, *MODULE, *SQLPKG, *FILE (save file), or *CMD.

Length of object path name
     INPUT; BINARY(4)

The length of the object path name. If the format of object path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of object path name is OBJN0100, only the path name itself is included.

Format of object path name
     INPUT; CHAR(8)

The format of the object path name parameter

OBJN0100	The object path name is a simple path name.
OBJN0200	The object path name is an LG-type path name.

Receiver
     OUTPUT; CHAR(*)

The structure that returns one or more blocks of certificate information from a digitally signed object.

Length of receiver
     INPUT; BINARY(4)

Size (in bytes) of the receiver available for signatures to be returned.

Format of receiver
  INPUT; CHAR(8)

  The format of certificate fields returned in the receiver.

CERT0200 All certificate text fields are translated from the ASCII format into the job CCSID.
CERT0210 All certificate fields are returned in the original certificate ASCII format.

Error code
  I/O; CHAR(*)

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

Receiver Structure

The receiver structure is comprised of:

1. A header section
2. An array of subheader sections called signature sections
3. For each subheader section, a Certificate Format CERT0200 (or CERT0210) as documented in the Parse Certificate (QSYPARSEC, QsyParserCertificate) API.

Header

Receiver Header area

For a description of the fields, see "Field Descriptions" on page 107

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes_Returned
4	4	BINARY(4)	Bytes_Available
8	8	BINARY(4)	Offset_To_Sections
12	12	BINARY(4)	Length_Of_Section
16	10	BINARY(4)	Number_Of_Sections
20	14	BINARY(4)	Number_Signatures_Returned
24	18	BINARY(4)	Number_Signatures_Available
28	1C	BINARY(4)	Composite_Object
32	20	BINARY(4)	Version
36	24	BINARY(4)	IBM_Signed
40	28	CHAR(1)	Core Signed
41	29	CHAR(1)	Entire Signed
42	30	CHAR(1)	Compressed Signature Exists
43	31	CHAR(1)	Decompressed Signature Exists
44	2C	CHAR(24)	Reserved for future use

Field Descriptions

Bytes_Returned. Number of bytes returned by the API into the receiver.

Bytes_Available. Number of bytes available from the API

Offset_To_Sections. Offset from beginning of struct to the first signature section

Length_Of_Section. Length of an individual signature section

Number_Of_Sections. The number of signature sections in the array of signature sections

Number_Signatures_Returned. How many signatures were returned

Number_Signatures_Available. How many signatures were available

Composite_Object. Composite object indicator. 0 if not composite; nonzero if composite.

Version. V5R1 value is zero. V5R2 value of 1 added to indicate added fields.

IBM_Signed. Whether IBM i5/OS signed. 1 if IBM i5/OS signed.

Core Signed. If Version is 0, Reserved.
″Core″ is applicable to *CMD objects only.
'1' if there is a ″Core″ signature for some certificate. '0' if there is no ″Core″ signature on the object.

Entire Signed. If Version is 0, Reserved.
'1' if there is an ″Entire″ signature for some certificate. '0' if there is no ″Entire″ signature on the object.

Compressed Signature Exists. If Version is 0, Reserved.
'1' indicates the object has a digital signature for the compressed object for some certificate. '0' indicates the object has no digital signature for the compressed object.

Decompressed Signature Exists. If Version is 0, Reserved.
'1' indicates the object has a digital signature for the decompressed object for some certificate. '0' indicates the object has a digital signature for the decompressed object.

Signature Section

For a description of the fields, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset_Cert_Info
4	4	BINARY(4)	Length_Cert_Info
8	8	CHAR(8)	Certificate_Format
16	10	CHAR(1)	Reserved1
17	11	CHAR(7)	Parse_Msg_ID
24	18	CHAR(14)	Date_Signed
38	26	CHAR	Signature_Scope
39	27	CHAR	Compressed_Signature;

Offset			
Dec	Hex	Type	Field
40	28	CHAR	Decompressed_Signature;
41	29	CHAR(23)	Reserved2

Field Descriptions

Offset_Cert_Info. Offset from beginning of receiver to the certificate information

Length_Cert_Info. Length of the certificate information

Certificate_Format. Format of the parsed certificate. Format is CERT0210 or CERT0200 per input request or CERT0000 if not parsed.

Reserved1. Reserved byte

Parse_Msg_ID. Message result, if any, from parsing the certificate

Date_Signed. YYYYMMDDhhmmss format where YYYY represents the year, MM the month, hh the hour, mm the minutes, and ss the seconds.

Signature_Scope. If Version is 0, Reserved.
'E' if there is an "Entire" signature for some certificate. 'C' if there is a "Core" signature on the object.

Compressed_Signature;. If Version is 0, Reserved.
'1' indicates the object has a digital signature for the compressed object for this certificate. '0' indicates the object has no digital signature for the compressed object for this certificate.

Decompressed_Signature;. If Version is 0, Reserved.
'1' indicates the object has a digital signature for the decompressed object for this certificate. '0' indicates the object has no digital signature for the decompressed object for this certificate.

Reserved2. RESERVED bytes

Certificate Format CERT0200 (or CERT0210)

Each subheader section provides a receiver-start relative offset to a certificate format CERT0200 (or CERT0210) as documented in the Parse Certificate (QSYPARSEC, QsyParserCertificate) API.

The certificate format has offsets relative to a beginning offset of its own structure under the heading "Certificate Format CERT0200 (Plain Text)" in the API for Parse Certificate. These are retained in the API. These structure offsets are thus displacements relative to the certificate format beginning within the receiver.

If a message is issued when using the interface to parse the certificate, the message ID will be copied into the signature section (the subheader) field Parse_Msg_ID.

Error Messages

Message ID	Error Message Text
CPFA0A9 E	Object not found.
CPFB720 E	Object type does not support signing.
CPFB722 E	Object not signed.

Message ID	Error Message Text
CPFB724 E	Option &2 of the operating system is required to work with object signatures.
CPFB735 E	The digital signing API parameter &1 is not large enough.
CPFB736 E	The digital signing API parameter &1 is not small enough.
CPFB737 E	The digital signing API parameter &1 is a null pointer.
CPFB738 E	The digital signing API parameter &1 is not a valid format type.
CPFB739 E	The digital signing API parameter &1 is out of range.
CPFB740 E	The format name for the pathname is not valid.
CPFB741 E	The length of the path name parameter is not valid.
CPFB742 E	The subdirectory option is not a valid value.
CPFB743 E	The value for stopping on the first error is not valid.
CPFB745 E	The format name for the results file path name is not valid.
CPFB746 E	The results file path name length is not large enough.
CPFB749 E	Object signature operation ended abnormally. &3 objects attempted, &2 objects successfully processed.

API introduced: V5R1

Retrieve Objects Secured by Authorization List (QGYRATLO) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	List information	Output	Char(80)
4	Section information	Output	Char(64)
5	Number of records to return	Input	Binary(4)
6	Format name	Input	Char(8)
7	Authorization list	Input	Char(10)
8	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Retrieve Objects Secured by Authorization List (QGYRATLO) API provides a list of objects that are secured by an authorization list. This API provides information similar to the Display Authorization List Objects (DSPAUTLOBJ) command and the List Objects Secured by Authorization List (QSYLATLO) API.

Differences between QSYLATLO and QGYRATLO

The QGYRATLO API returns the same information that the List Objects Secured by Authorization List (QSYLATLO) API provides, but takes a complete snapshot at once and allows subsequent records to be obtained through the Get List Entries (QGYGTLE) API.

Authorities and Locks

Authorization List Authority
        Must not be *EXCLUDE authority

Required Parameter Group

Receiver variable
        OUTPUT; CHAR(*)

The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. For formats that contain variable length data, the receiver variable length must be large enough to hold the fixed portion of the record.

List information
OUTPUT; CHAR(80)

The variable that is used to return status information about the list of secured objects that were opened. See "Format of List Information" on page 114 for a description of this parameter.

Section information
OUTPUT; CHAR(64)

The variable that is used to return entry information about the list of secured objects that was opened. See "Format of Section Information" on page 116 for a description of this parameter.

Number of records to return
INPUT; BINARY(4)

The number of records in the list to put into the receiver variable after the entire list has been built. If -1 is specified, then all the records will be returned.

Format name
INPUT; CHAR(8)

The name of the format that is used to list objects secured by the authorization list.

You can specify these formats:

"ATLO0100 Format" on page 111	Each entry contains the object name, library, type, authority holder indicator, auxiliary storage pool (ASP) device name of library, and ASP device name of object.
"ATLO0110 Format" on page 111	This format only returns path names for objects in a directory. Each entry contains the offset to the path name, the length of the path name, type, authority holder indicator, ASP device name of object, and the path name value. Objects in the QSYS and QDLS file systems are not returned with this format.
"ATLO0200 Format" on page 111	Each entry contains the same information as ATLO0100 plus the object owner, attribute, text, and primary group.
"ATLO0210 Format" on page 112	This format only returns path names for objects in a directory. Each entry contains the same information as format ATLO0110 plus the object owner, attribute, text, and primary group. Objects in the QSYS and QDLS file systems are not returned with this format.
"ATLO0300 Format" on page 112	Each entry contains the length of the entry, object name, library, type, authority holder indicator, document library object (DLO) name, the name of the folder that the DLO is in, the displacement to the path name, the length of the path name, ASP device name of library, ASP device name of object, and the path name value. Objects in all file systems will be returned with this format.
"ATLO0400 Format" on page 113	Each entry contains the same information as ATLO0300 plus the object owner, primary group, attribute, and text. Objects in all file systems are returned with this format.

Authorization list
INPUT; CHAR(10)

The name of the authorization list for which the secured objects are returned.

Error code
> I/O; CHAR(*)

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

Format of Receiver Variable

The following tables describe the order and format of the data returned in the receiver variable. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 113.

ATLO0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(10)	ASP device name of library
41	29	CHAR(10)	ASP device name of object

ATLO0110 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(1)	Reserved
20	14	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

ATLO0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Authority holder
31	1F	CHAR(10)	Owner
41	29	CHAR(10)	Attribute
51	33	CHAR(50)	Text description
101	65	CHAR(10)	Primary group

Offset			
Dec	Hex	Type	Field
111	6F	CHAR(10)	ASP device name of library
121	79	CHAR(10)	ASP device name of object

ATLO0210 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset to path name
4	4	BINARY(4)	Length of path name
8	8	CHAR(10)	Object type
18	12	CHAR(1)	Authority holder
19	13	CHAR(10)	Owner
29	1D	CHAR(10)	Attribute
39	27	CHAR(50)	Text description
89	59	CHAR(10)	Primary group
99	63	CHAR(1)	Reserved
100	64	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

ATLO0300 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of entry
4	4	CHAR(10)	Object name
14	0E	CHAR(10)	Library name
24	18	CHAR(10)	Object type
34	22	CHAR(1)	Authority holder
35	23	CHAR(12)	DLO name
47	2F	CHAR(63)	Folder name
110	6E	CHAR(2)	Reserved
112	70	BINARY(4)	Displacement to path name
116	74	BINARY(4)	Length of path name
120	78	CHAR(10)	ASP device name of library
130	82	CHAR(10)	ASP device name of object
		CHAR(*)	Path name

ATLO0400 Format

Offset Dec	Offset Hex	Type	Field
0	0	BINARY(4)	Length of entry
4	4	CHAR(10)	Object name
14	0E	CHAR(10)	Library name
24	18	CHAR(10)	Object type
34	22	CHAR(1)	Authority holder
35	23	CHAR(12)	DLO name
47	2F	CHAR(63)	Folder name
110	6E	CHAR(2)	Reserved
112	70	BINARY(4)	Displacement to path name
116	74	BINARY(4)	Length of path name
120	78	CHAR(10)	Owner
130	82	CHAR(10)	Attribute
140	8C	CHAR(50)	Text description
190	BE	CHAR(10)	Primary group
200	C8	CHAR(10)	ASP device name of library
210	D2	CHAR(10)	ASP device name object
		CHAR(*)	Path name

Field Descriptions

ASP device name of library. The auxiliary storage pool (ASP) device name where the object's library is stored. If the object's library is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

ASP device name of object. The auxiliary storage pool (ASP) device name where the object is stored. If the object is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

Attribute. The attribute of the secured object. If the object is not in the QSYS or QDLS file system, this field is blank.

Authority holder. Whether the object is an authority holder. If the object is an authority holder, this field is Y. If not, this field is N.

Authorization list. The name of the authorization list for which the list of objects is returned.

Authorization list library name. The name of the library that contains the authorization list.

Displacement to path name. The displacement in the entry to the start of the path name. The displacement will be set to zero if the receiver variable is not large enough to hold the path name.

DLO name. The document library object (DLO) name for the object. If the object is not a *DOC (document) or *FLR (folder) object, this field is blank.

Folder name. The name of the folder that contains the DLO object. If the object is not in a folder, this field contains *NONE.

Length of entry. The length (in bytes) of the current entry.

Length of path name. The length (in bytes) of the path name.

Library name. The name of the library that contains the object.

Object name. The name of the object that is secured by the authorization list. If the object is not in the QSYS or QDLS file system, this field is blank.

Object type. The type of secured object.

Offset to path name. The offset in the receiver variable to the start of the path name. The offset will be set to zero if the receiver variable is not large enough to hold the path name.

Owner. The name of the owner of the authorization list or object.

Path name. The path name of the object that is secured by the authorization list. The user must request a format that supports path names if path names are to be included in the information that is returned in the receiver variable. The structure of the path name returned follows:

Description	Type
CCSID of the returned path name	Binary(4)
Country or region ID	Char(2)
Language ID	Char(3)
Reserved field	Char(3)
Flag byte	Binary(4)
Number of bytes in the path name	Binary(4)
Path delimiter	Char(2)
Reserved field	Char(10)
Path name value	Char(*)

Primary group. The name of the user who is the primary group for the authorization list or object. If there is no primary group for the authorization list or object, this field contains a value of *NONE.

Reserved. An ignored field.

Text description. The descriptive text for the secured object. If the object is not in the QSYS or QDLS file system, this field is blank.

Format of List Information

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Total records
4	4	BINARY(4)	Records returned
8	8	CHAR(4)	Request handle
12	C	BINARY(4)	Record length
16	10	CHAR(1)	Information complete indicator
17	11	CHAR(13)	Date and time created
30	1E	CHAR(1)	List status indicator
31	1F	CHAR(1)	Reserved
32	20	BINARY(4)	Length of information returned

Offset		Type	Field
Dec	Hex		
36	24	BINARY(4)	First record in buffer
40	28	BINARY(4)	Reason code
44	2C	CHAR(36)	Reserved

Field Descriptions

Date and time created. The date and time when the list was created.

The 13 characters are:

0	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
2-7	The date, in YYMMDD (year, month, and day) format.
8-13	The time of day, in HHMMSS (hours, minutes, and seconds) format.

First record in buffer. The number of the first record in the receiver variable.

Information complete indicator. Whether all requested information has been supplied. Possible values follow:

I	Incomplete information. An interruption causes the list to contain incomplete information about a buffer or buffers.
P	Partial and accurate information. Partial information is returned when the maximum space was used and not all of the buffers requested were read.
C	Complete and accurate information. All the buffers requested are read and returned.

Length of information returned. The size, in bytes, of the information that is returned in the receiver variable.

List status indicator. The status of building the list.

Possible values follow:

2	The list has been completely built.

Reason code. The reason code that further describes why the list is only a subset of all objects. The following values can be returned:

- Reason code 0000. The list returned in the receiver variable contains all objects that meet the search criteria.
- Reason code 0001. Objects were found that meet the search criteria, but they could not be included in the returned list. The requested format could not handle path names for directory objects.
- Reason code 0002. Objects were found that meet the search criteria, but they could not be included in the returned list. The requested format could not handle objects found in library QSYS.
- Reason code 0003. Directory objects were found, but they did not have links to them.

Record length. The length of each record of information returned. For variable length records, this value is set to zero. For variable length records you can obtain the length of individual records from the records themselves.

Records returned. The number of records returned in the receiver variable. This is the smallest of the following three values:

- The number of records that will fit into the receiver variable.
- The number of records in the list.
- The number of records that are requested.

Request handle. The handle of the request that can be used for subsequent requests of information from the list. The handle is valid until the Close List (QGYCLST) API is called to close the list, or until the job ends.

Note: This field should be treated as a hexadecimal field. It should not be converted from one CCSID to another, for example, EBCDIC to ASCII, because doing so could result in an unusable value.

Reserved. An ignored field.

Total records. The total number of records available in the list.

Format of Section Information

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Entry number of first QSYS.LIB object
4	4	BINARY(4)	Number of QSYS.LIB objects
8	8	BINARY(4)	Entry number of first QDLS object
12	0C	BINARY(4)	Number of QDLS objects
16	10	BINARY(4)	Entry number of first directory object
20	14	BINARY(4)	Number of directory objects
24	18	CHAR(40)	Reserved

Field Descriptions

Entry number of first directory object. The entry number of the first directory object (objects not in the QSYS.LIB or QDLS file system) that was returned in the receiver variable. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of directory objects field is 0, this value is also 0.

Entry number of first QDLS object. The entry number of the first QDLS object that was returned in the receiver variable. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of QDLS objects field is 0, this value is also 0.

Entry number of first QSYS.LIB object. The entry number of the first QSYS.LIB object that was returned in the receiver variable. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If the number of QSYS.LIB objects field is 0, this value is also 0.

Number of directory objects. The number of objects in directories (objects not in the QSYS.LIB or QDLS file system) that were returned in the receiver variable. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If there are no entries for objects in directories in the receiver variable, 0 is returned.

Number of QDLS objects. The number of objects in the QDLS file system that were returned in the receiver variable. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If there are no entries for QDLS objects in the receiver variable, 0 is returned.

Number of QSYS.LIB objects. The number of objects in the QSYS.LIB file system that were returned in the receiver variable. This value is only set if you are using format ATLO0300 or ATLO0400. Otherwise, -1 is returned. If there are no entries for QSYS.LIB objects in the receiver variable, 0 is returned.

Reserved. An ignored field.

Error Messages

Message ID	Error Message Text
CPF22AF E	Not authorized to authorization list &1.
CPF2283 E	Authorization list &1 does not exist.
CPF2289 E	Unable to allocate authorization list &1.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R1

Retrieve Security Attributes (QSYRTVSA) API

Required Parameter Group:

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

Threadsafe: No

The Retrieve Security Attributes (QSYRTVSA) API retrieves information about the current and pending security attributes of the system.

Authorities and Locks

None.

Required Parameter Group

Receiver variable
> OUTPUT; CHAR(*)

> The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format name
> INPUT; CHAR(8)

> The format of the command information to be returned.

> The following format name may be used:

"RTSA0100 Format"   Basic system security attributes.

Error code
> I/O; CHAR(*)

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

RTSA0100 Format

The following table describes the information that is returned in the receiver variable for the RTSA0100 format. For detailed descriptions of the fields, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Current security level
12	C	BINARY(4)	Pending security level
16	10	BINARY(4)	Current password level
20	14	BINARY(4)	Pending password level
24	18	CHAR(1)	Allow change to security related system values
25	19	CHAR(1)	Allow add of digital certificates
26	1A	CHAR(1)	Allow service tools user ID password change

Field Descriptions

Allow add of digital certificates. Specifies whether or not digital certificates can be added to a certificate store using the Add Verifier (QYDOADDV) API, and whether or not the password for a certificate store can be reset using Digital Certificate Manager (DCM).

0	Digital certificates cannot be added to a certificate store using the QYDOADDV API, and certificate store passwords cannot be reset using DCM.
1	Digital certificates can be added to a certificate store using the QYDOADDV API, and certificate store passwords can be reset using DCM.

Allow change to security related system values. Specifies whether or not the security related system values can be changed.

0	The security related system values cannot be changed.
1	The security related system values can be changed.

Allow service tools user ID password change. Specifies whether or not a service tools user ID with a default password that is expired can change its own password.

0	A service tools user ID with a default password that is expired cannot change its own password.
1	A service tools user ID with a default password that is expired can change its own password.

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.

Current security level. This is the security level that is currently being used by the system. See the QSECURITY system value for a list of the possible values.

Current password level. This is the password level that is currently being used by the system. See the QPWDLVL system value for a list of the possible values.

Pending security level. This is the security level that the system will use after the next IPL. It is the same value as displayed by the Display System Value (DSPSYSVAL) command for the QSECURITY system value. See the QSECURITY system value for a list of the possible values.

Pending password level. This is the password level that the system will use after the next IPL. It is the same value as displayed by the Display System Value (DSPSYSVAL) command for the QPWDLVL system value. See the QPWDLVL system value for a list of the possible values.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

Retrieve User Application Information (QsyRetrieveUserApplicationInfo) API

≫

Syntax for QsyRetrieveUserApplicationInfo:

```
#include <qsyusrin.h>

void QsyRetrieveUserApplicationInfo
      (void          *Receiver_variable,
       int           *Length_of_receiver_variable,
       void          *Return_records_feedback_information,
       char          *Format_name,
       char          *User_profile,
       char          *Application_information_ID,
       int           *Length_of_application_information_ID,
       void          *Error_code);
```

Service Program: QSYUSRIN
Default Public Authority: *USE
Threadsafe: No

≪

The Retrieve User Application Information (QsyRetrieveUserApplicationInfo) API returns a list of application information entries for a user profile.

Authorities and Locks

User Profile Authority
        READ

Required Parameter Group

Receiver variable
        OUTPUT; CHAR(*)

The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
        INPUT; BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable.

Returned records feedback information
        OUTPUT; CHAR(12)

Information about the entries that are returned in the receiver variable.

See "Format of Returned Records Feedback Information" on page 122 for details.

Format name
        INPUT; CHAR(8)

The name of the format that is used to retrieve application information entries for the user profile.

You can specify this format:

RUAI0100   For a detailed description of this format, see "RUAI0100 Format."

User profile
> INPUT; CHAR(10)

> The name of the user profile for which the application information will be retrieved. The special value *CURRENT may be specified to retrieve application information for the user profile that calls this API.

Application information ID
> INPUT; CHAR(*)

> The ID for the application information to retrieve. The following can be specified for the application information ID:

> **generic***
>> All application information IDs that have IDs beginning with the generic string will be retrieved.

> **application information ID**
>> Specific application information ID will be retrieved.

Length of application information ID
> INPUT; BINARY(4)

> The length of the application information ID. The length of the application information ID may be from 1 to 200.

Error code
> I/O; CHAR(*)

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

Receiver Variable Description

The following tables describe the order and format of the data returned in the receiver variable. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 122.

RUAI0100 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of entry
4	4	CHAR(200)	Application information ID
204	CC	BINARY(4)	Displacement to user application information
208	D0	BINARY(4)	Length of user application information
212	D4	BINARY(4)	CCSID of user application information
216	D8	CHAR(6)	First valid release
		CHAR(*)	User application information

Format of Returned Records Feedback Information

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of user application information entries

Field Descriptions

Application information ID. The application information ID that identifies this entry.

Bytes available. The number of bytes of data available to be returned to the user in the receiver variable. If all data is returned, bytes available is the same as the number of bytes returned. If the receiver variable was not large enough to contain all of the data, this value is estimated based on the total number of application information entries for the user profile and the format specified.

Bytes returned. The number of bytes of data returned to the user in the receiver variable. This is the lesser of the number of bytes available to be returned or the length of the receiver variable.

CCSID of user application information. The CCSID of the user application information. This will be the default job CCSID of the job that last updated the user application information.

Displacement to user application information. The displacement in the entry to the start of the user application information.

First valid release. The first release that this application information is valid. This field will be in the format VxRxMx (for example, V5R3M0).

Length of entry. The length (in bytes) of the current entry. This length can be used to access the next entry.

Length of user application information. The length (in bytes) of the user application information.

Number of user application information entries. The number of complete entries returned in the list of user application information entries. A value of zero is returned if the list is empty.

User application information. The user application information that is associated with the user profile.

Error Messages

Message ID	Error Message Text
CPFA0AA E	Error occurred while attempting to obtain space.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
CPF2222 E	Storage limit is greater than specified for user profile &1.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R3

Retrieve User Authority to Object (QSYRUSRA) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Receiver variable length	Input	Binary(4)
3	Format name	Input	Char(8)
4	User profile name	Input	Char(10)
5	Qualified object name	Input	Char(20)
6	Object type	Input	Char(10)
7	Error code	I/O	Char(*)

Optional Parameter Group 1:

8	ASP device	Input	Char(10)

Optional Parameter Group 2:

9	Path name	Input	Char(*)
10	Length of Path name	Input	Binary(4)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve User Authority to Object (QSYRUSRA) API returns a specific user's authority for an object to the caller.

Authorities and Locks

The following authorities are required for the user calling this API, unless the user profile specified is *CURRENT, the caller owns the object, or the object is an authorization list:

- *OBJMGT authority to the object specified.
- *READ authority to the user profile specified (unless *PUBLIC is specified).
- *USE authority to the Auxiliary Storage Pool Device.

If previous programs in the program stack adopt their owner's authority, the adopted authority for the current program is the accumulated adopted authority from all other programs in the program stack that adopt authority. Adopted authority is only valid when the user specified is *CURRENT. If a path name is specified, adopted authority is not used when accessing the path object.

If a path name is specified, *X authority is required for each directory in the path.

If a path name is specified, adopted authority is not used to locate the object but adopted authority will be used when authority information is retrieved for the object.

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The variable used to return the user's authority to the object. This variable must be at least 8 bytes long.

Receiver variable length
> INPUT; BINARY(4)

> The length of the receiver variable. The variable must be at least 8 bytes long.

Format name
> INPUT; CHAR(8)

> The name of the format used to return the authority information.

> You can specify the following special value:

"USRA0100 Format" on page 125　　　All authority information is returned.

User profile name
> INPUT; CHAR(10)

> The name of the user whose object authority is returned.

> You can specify the following special values:

*CURRENT　　　The authority of the user currently running to the specified object is returned.
*PUBLIC　　　The public authority for the object is returned.

Qualified object name
> INPUT; CHAR(20)

> The name of the object whose authority is returned. The first 10 characters specify the object name, and the second 10 characters specify the library.

> If you want to use a path name instead of a qualified object name, then use this special value for the object name:

*OBJPATH　　　Use the optional parameters, path name and path name length, to specify the object name. When this special value is specified, the library name must be blanks.

> You can use these special values for the library name:

*CURLIB　　　The current library is used to locate the object. If there is no current library, QGPL (general purpose library) is used.
*LIBL　　　The library list is used to locate the object.

Object type
> INPUT; CHAR(10)

> The type of object for which authority information is returned. The object type must be blank if *OBJPATH is specified for the qualified object name.

Error code
> I/O; CHAR(*)

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

Optional Parameter Group 1

ASP device
> INPUT; CHAR(10)

> The name of the auxiliary storage pool (ASP) device in which to search for the library that contains the object. If *OBJPATH is specifed for the qualifed object name, the ASP device must be *.

The valid values are:

*	All ASPs associated with the job will be searched. This is the default value if the parameter is not specified.
SYSBAS	The system ASP and all basic user ASPs will be searched.
ALL	All ASPs that are currently available will be searched.
ASP device name	The specified ASP will be searched.

If *CURLIB or *LIBL is specified for the library then the ASP device parameter must be specified as *.

Optional Parameter Group 2

Path name
INPUT; CHAR(*)

The object name, specified as a path name. This parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

If the length of the path name is -1, then this parameter is assumed to be a Qlg_Path_Name_T structure that contains a path name or a pointer to a path name. For more information on the Qlg_Path_Name_T structure, see Path name format.

Length of path name
INPUT; BINARY(4)

The length of the path name in bytes. If the length is -1, the path name parameter is assumed to be a Qlg_Path_name_T structure. This value must be zero if no path name is specified.

Receiver Variable Description

The following tables describe the order and format of the data returned in the receiver variable. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 126.

USRA0100 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Object authority / Data authority
18	12	CHAR(1)	Authorization list management
19	13	CHAR(1)	Object operational
20	14	CHAR(1)	Object management
21	15	CHAR(1)	Object existence
22	16	CHAR(1)	Data read
23	17	CHAR(1)	Data add
24	18	CHAR(1)	Data update
25	19	CHAR(1)	Data delete
26	1A	CHAR(10)	Authorization list
36	24	CHAR(2)	Authority source
38	26	CHAR(1)	Some adopted authority

Offset		Type	Field
Dec	Hex		
39	27	CHAR(10)	Adopted object authority
49	31	CHAR(1)	Adopted authorization list management
50	32	CHAR(1)	Adopted object operational
51	33	CHAR(1)	Adopted object management
52	34	CHAR(1)	Adopted object existence
53	35	CHAR(1)	Adopted data read
54	36	CHAR(1)	Adopted data add
55	37	CHAR(1)	Adopted data update
56	38	CHAR(1)	Adopted data delete
57	39	CHAR(1)	Adopted data execute
58	3A	CHAR(10)	Reserved
68	44	CHAR(1)	Adopted object alter
69	45	CHAR(1)	Adopted object reference
70	46	CHAR(10)	Reserved
80	50	CHAR(1)	Data execute
81	51	CHAR(10)	Reserved
91	5B	CHAR(1)	Object alter
92	5C	CHAR(1)	Object reference
93	5D	CHAR(10)	ASP device name of library
103	67	CHAR(10)	ASP device name of object
113	71	CHAR(3)	Reserved
116	74	BINARY(4)	Offset to group information table
120	78	BINARY(4)	Number of group table entries returned
*	*	Char(*)	Group information table repeated for each of the user's groups

Field Descriptions

Adopted authorization list management. Whether the user has adopted this authority to the object. If the user adopted the authority, this field is Y. If not, this field is N.

Adopted data add. Whether the user has adopted this authority to the object. If the user has adopted the authority, this field is Y. If not, this field is N.

Adopted data delete. Whether the user has adopted this authority to the object. If the user has adopted the authority, this field is Y. If not, this field is N.

Adopted data execute. Whether the user has adopted this authority to the object. If the user adopted the authority, this field is Y. If not, this field is N.

Adopted data read. Whether the user has adopted this authority to the object. If the user has adopted the authority, this field is Y. If not, this field is N.

Adopted data update. Whether the user has adopted this authority to the object. If the user has adopted the authority, this field is Y. If not, this field is N.

Adopted object alter. Whether the user has adopted this authority to the object. If the user adopted the authority, this field is Y. If not, this field is N.

Adopted object authority. The user's adopted authority to the object. This field is only valid if some of the user's authority is adopted. If the user does not adopt authority, this field will be blank.

If a qualified object name is specified, the possible values are:

*ALL	The user adopted all object (operational, management, existence, alter, and reference) and data (read, add, update, delete, and execute) authorities to the object.
*CHANGE	The user adopted object operational and all data authorities to the object.
*USE	The user adopted object operational and data read and execute authorities to the object.
USER DEF	The user adopted some combination of object and data authorities that do not relate to a special value. The individual authorities for the user should be checked to determine what authority the user has adopted to the object.

If a path name is specified, the possible values are:

*RWX	The user has object operational and all data authorities.
*RW	The user has object operational and data read, data add, data update, and data delete authorities.
*RX	The user has object operational, data read, and data execute authorities
*WX	The user has object operational and data add, data update, data delete, and data execute authorities.
*R	The user has object operational and data read authorities.
*W	The user has object operational and data add, data update, and data delete authorities.
*X	The user has object operational and data execute authorities.
*EXCLUDE	The user has exclude authority.
*NONE	The user does not have object operational or any data authorities.
USER DEF	The user has some combination of object and data authorities that do not relate to a special value. The individual authorities for the user should be checked to determine what authority the user has to the object.

Note: If *OBJPATH is specified for the qualified object name parameter, be aware that adopted authority is not used by most commands and APIs that accept path names as input.

Adopted object existence. Whether the user adopted this authority to the object. If the user adopted the authority, this field is Y. If not, this field is N.

Adopted object management. Whether the user has adopted this authority to the object. If the user has adopted the authority, this field is Y. If not, this field is N.

Adopted object operational. Whether the user has adopted this authority to the object. If the user has adopted the authority, this field is Y. If not, this field is N.

Adopted object reference. Whether the user has adopted this authority to the object. If the user adopted the authority, this field is Y. If not, this field is N.

ASP device name of library. The auxiliary storage pool (ASP) device name where the object's library is stored. If the object's library is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

ASP device name of object. The auxiliary storage pool (ASP) device name where the object is stored. If the object is in the system ASP or one of the basic user ASPs, this field contains *SYSBAS.

Authority source. Indicates where the authority that the user has to the object initially came from. The authority may be a combination of authority from this source plus adopted authority.

This field contains one of the following special values:

UA	The user has *ALLOBJ special authority.
UO	The user is privately authorized to the object.
UL	The user is privately authorized to the authorization list securing the object.
GA	The user's groups have *ALLOBJ special authority.
GO	The user's groups are privately authorized to the object.
GL	The user's groups are privately authorized to the authorization list securing the object.
GC	The user's groups have a combination of private authority to the object and private authority to the authorization list securing the object.
PO	The user accesses the object through the public authority.
PL	The user accesses the object through the public authority on the authorization list securing the object.
AD	All of the authority that the user has comes from adopted authority. This value is only returned if the user is *CURRENT.

Authorization list. The name of the authorization list securing the object.

This field can contain one of the following special values:

*NONE	There is no authorization list securing the object.
*DAMAGED	The authorization list securing the object is damaged.

Authorization list management. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Bytes available. The number of bytes of data available to be returned to the user. If all data is returned, this is the same as the number of bytes returned. If the receiver variable was not big enough to contain all of the data, this is the number of bytes that can be returned.

Bytes returned. The number of bytes of data returned to the user. This is the lesser of the number of bytes available to be returned or the length of the receiver variable.

Data add. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data delete. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data execute. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data read. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Data update. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Group information table. A list of the user's group authorities to the object.

Number of group table entries returned. Number of group table entries returned in the receiver variable.

Object alter. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object authority / Data authority. If a qualified object name is specifed, this is a special value indicating the user's total authority to the object including adopted authority (if the user is *CURRENT). If a path name is specified, this is a special value indicating the user's data authority to the object and includes any adopted authority (if the user is *CURRENT).

If a qualified object name is specified, the possible values are:

*ALL	The user has all object (operational, management, existence, alter and reference) and data (read, add, update, delete, and execute) authorities to the object.
*CHANGE	The user has object operational and all data authorities to the object.
*USE	The user has object operational, data read, and data execute authorities to the object.
*EXCLUDE	The user has exclude authority to the object.
USER DEF	The user has some combination of object and data authorities that do not relate to a special value. The individual authorities for the user should be checked to determine what authority the user has to the object.

If a path name is specified, the possible values are:

*RWX	The user has object operational and all data authorities.
*RW	The user has object operational and data read, data add, data update, and data delete authorities.
*RX	The user has object operational, data read, and data execute authorities
*WX	The user has object operational and data add, data update, data delete, and data execute authorities.
*R	The user has object operational and data read authorities.
*W	The user has object operational and data add, data update, and data delete authorities.
*X	The user has object operational and data execute authorities.
*EXCLUDE	The user has exclude authority.
*NONE	The user does not have object operational or any data authorities.
USER DEF	The user has some combination of object and data authorities that do not relate to a special value. The individual authorities for the user should be checked to determine what authority the user has to the object.

Object existence. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object management. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object operational. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Object reference. Whether the user has this authority to the object. If the user has the authority, this field is Y. If not, this field is N.

Offset to group information table. Offset from the beginning of the receiver variable to the first group table entry.

Reserved. An ignored field set to hexadecimal zeros.

Some adopted authority. Whether some of the authority that the user has to the object comes from adopted authority. If some of the authority is adopted, this field is Y. If not, this field is N. This field can only contain Y if the user is *CURRENT.

Group Information Table

This table holds information about the authorities a group has to the object.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Displacement to next group entry
4	4	CHAR(10)	Group profile
14	E	CHAR(10)	Object authority / Data authority
24	18	CHAR(1)	Authority source
25	19	CHAR(1)	Authorization List Management
26	1A	CHAR(1)	Object operational
27	1B	CHAR(1)	Object management
28	1C	CHAR(1)	Object existence
29	1D	CHAR(1)	Object alter
30	1E	CHAR(1)	Object reference
31	1F	CHAR(10)	Reserved
41	29	CHAR(1)	Data read
42	2A	CHAR(1)	Data add
43	2B	CHAR(1)	Data update
44	2C	CHAR(1)	Data delete
45	2D	CHAR(1)	Data execute
46	2E	CHAR(2)	Reserved

Field Descriptions

Authority source. Where the group's authority comes from. The value of this field is one of these special values:

A	The group has *ALLOBJ special authority.
O	The group authority comes from private authority to the object.
L	The group authority comes from the authorization list securing the object.
blank	There is no authority source for the group.

Authorization List Management. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Data add. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Data delete. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Data execute. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Data read. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Data update. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Displacement to next group entry. Displacement to the next group entry. This field is 0 if there is not another group entry.

Group profile. Name of a group in the user's profile.

Object alter. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Object authority / Data authority. If a qualified object name is specified, this is a special value indicating the group's authority to the object. If a path name is specified, this is a special values indications the group's data authority to the object.

If a qualified object name is specified, this is one of the following values:

*ALL	The group has all object (operational, management, existence, alter and reference) and data (read, add, update, delete, and execute) authorities to the object.
*CHANGE	The group has object operational and all data authorities to the object.
*USE	The group has object operational, data read, and data execute authorities to the object.
*EXCLUDE	The group has exclude authority to the object, or authorization list management authority.
USER DEF	The group has some combination of object and data authorities that do not relate to a special value. The individual authorities for the group should be checked to determine what authority the group has to the object.

If a path name is specified, this is one of the following values:

*RWX	The user has object operational and all data authorities.
*RW	The user has object operational and read, add, update, and delete data authorities.
*RX	The user has object operational, data read, and data execute authorities
*WX	The user has object operational and add, update, delete, and execute data authorities.
*R	The user has object operational and data read authorities.
*W	The user has object operational and add, update, and delete data authorities.
*X	The user has object operational and data execute authorities.
*EXCLUDE	The user has exclude authority.
*NONE	The user does not have object operational or any data authorities.
USER DEF	The group has some combination of object and data authorities that do not relate to a special value. The individual authorities for the group should be checked to determine what authority the group has to the object.

Object existence. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Object management. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Object operational. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Object reference. Whether the group has this authority to the object. If the group has the authority, this field is Y. If not, this field is N.

Error Messages

Message ID	Error Message Text
CPF18A2 D	Path name parameters not specified.
CPF2203 E	User profile &1 not correct.

Message ID	Error Message Text
CPF2225 E	Not able to allocate internal system object.
CPF3CF1 E	Error code parameter not valid.
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.
CPF3C31 E	Object type &1 is not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8122 E	&8 damage on library &4.
CPF980B E	Object &1 in library &2 not available.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9814 E	Device &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9825 E	Not authorized to device &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF9873 E	ASP status is preventing access to object.
CPFA09C E	Not authorized to object.
CPFA09E E	Object in use. Object is &1.
CPFA0A3 E	Path name resolution causes looping.
CPFA0A7 E	Path name too long.
CPFA0A9 E	Object not found.
CPFA0AB E	Operation failed for object. Object is &1.
CPFA0AD E	Function not supported by file system.

API Introduced: V2R2

Retrieve User Information (QSYRUSRI) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Receiver variable length	Input	Binary(4)
3	Format name	Input	Char(8)
4	User profile name	Input	Char(10)
5	Error Code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve User Information (QSYRUSRI) API provides information about a user profile. This API provides information similar to the Retrieve User Profile (RTVUSRPRF) command or the Display User Profile (DSPUSRPRF) command when *BASIC is specified for the type parameter.

Authorities and Locks

User Profile Authority
>   *READ

Required Parameter Group

Receiver variable
>   OUTPUT; CHAR(*)

>   The variable used to return the information about the user. This variable must be at least 8 bytes long.

Receiver variable length
>   INPUT; BINARY(4)

>   The length of the receiver variable. The variable must be 8 bytes long.

Format name
>   INPUT; CHAR(8)

>   The name of the format used to return information about the user.

>   You can specify these formats:

"USRI0100 Format"	Sign-on and password information is returned. The password itself is not returned.
"USRI0200 Format" on page 134	Authority information is returned.
"USRI0300 Format" on page 134	All user information is returned.

User profile name
>   INPUT; CHAR(10)

>   The user name for which information is returned. You can specify the following special value:

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

Error code
>   I/O; CHAR(*)

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

Receiver Variable Description

The following tables describe the order and format of the data returned in the receiver variable. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 136.

USRI0100 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User profile name
18	12	CHAR(13)	Previous sign-on date and time
31	1F	CHAR(1)	Reserved
32	20	BINARY(4)	Sign-on attempts not valid

Offset			
Dec	**Hex**	**Type**	**Field**
36	24	CHAR(10)	Status
46	2E	CHAR(8)	Password change date
54	36	CHAR(1)	No password indicator
55	37	CHAR(1)	Reserved
56	38	BINARY(4)	Password expiration interval
60	3C	CHAR(8)	Date password expires
68	44	BINARY(4)	Days until password expires
72	48	CHAR(1)	Set password to expire
73	49	CHAR(10)	Display sign-on information
83	53	CHAR(1)	Local password management

USRI0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User profile name
18	12	CHAR(10)	User class name
28	1C	CHAR(15)	Special authorities
43	2B	CHAR(10)	Group profile name
53	35	CHAR(10)	Owner
63	3F	CHAR(10)	Group authority
73	49	CHAR(10)	Limit capabilities
83	53	CHAR(10)	Group authority type
93	5D	CHAR(3)	Reserved
96	60	BINARY(4)	Offset to array of supplemental groups
100	64	BINARY(4)	Number of supplemental groups
		ARRAY(*) OF CHAR(10)	Supplemental groups

USRI0300 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User profile name
18	12	CHAR(13)	Previous sign-on date and time

Offset			
Dec	Hex	Type	Field
31	1F	CHAR(1)	Reserved
32	20	BINARY(4)	Sign-on attempts not valid
36	24	CHAR(10)	Status
46	2E	CHAR(8)	Password change date
54	36	CHAR(1)	No password indicator
55	37	CHAR(1)	Reserved
56	38	BINARY(4)	Password expiration interval
60	3C	CHAR(8)	Date password expires
68	44	BINARY(4)	Days until password expires
72	48	CHAR(1)	Set password to expire
73	49	CHAR(10)	User class name
83	53	CHAR(15)	Special authorities
98	62	CHAR(10)	Group profile name
108	6C	CHAR(10)	Owner
118	76	CHAR(10)	Group authority
128	80	CHAR(10)	Assistance level
138	8A	CHAR(10)	Current library name
148	94	CHAR(10)	Initial menu name
158	9E	CHAR(10)	Initial menu library name
168	A8	CHAR(10)	Initial program name
178	B2	CHAR(10)	Initial program library name
188	BC	CHAR(10)	Limit capabilities
198	C6	CHAR(50)	Text description
248	F8	CHAR(10)	Display sign-on information
258	102	CHAR(10)	Limit device sessions
268	10C	CHAR(10)	Keyboard buffering
278	116	CHAR(2)	Reserved
280	118	BINARY(4)	Maximum allowed storage
284	11C	BINARY(4)	Storage used
288	120	CHAR(1)	Highest scheduling priority
289	121	CHAR(10)	Job description name
299	12B	CHAR(10)	Job description library name
309	134	CHAR(15)	Accounting code
324	144	CHAR(10)	Message queue name
334	14E	CHAR(10)	Message queue library name
344	158	CHAR(10)	Message queue delivery method
354	162	CHAR(2)	Reserved
356	164	BINARY(4)	Message queue severity
360	168	CHAR(10)	Output queue name
370	172	CHAR(10)	Output queue library name

Offset			
Dec	Hex	Type	Field
380	17C	CHAR(10)	Print device
390	186	CHAR(10)	Special environment
400	190	CHAR(10)	Attention-key-handling program name
410	19A	CHAR(10)	Attention-key-handling program library name
420	1A4	CHAR(10)	Language ID
430	1AE	CHAR(10)	Country or region ID
440	1B8	BINARY(4)	Character code set ID
444	1BC	CHAR(36)	User options
480	1E0	CHAR(10)	Sort sequence table name
490	1EA	CHAR(10)	Sort sequence table library name
500	1F4	CHAR(10)	Object auditing value
510	1FE	CHAR(64)	User action audit level
574	23E	CHAR(10)	Group authority type
584	248	BINARY(4)	Offset to array of supplemental groups
588	24C	BINARY(4)	Number of supplemental groups
592	250	BINARY(4)	User ID number
596	254	BINARY(4)	Group ID number
600	258	BINARY(4)	Offset to home directory
604	25C	BINARY(4)	Length of home directory
608	260	CHAR(16)	Locale job attributes
624	270	BINARY(4)	Offset to locale path name
628	274	BINARY(4)	Length of locale path name
632	278	CHAR(1)	Group member indicator
633	279	CHAR(1)	Digital certificate indicator
634	27A	CHAR(10)	Character identifier control
644	284	BINARY(4)	Offset to array of independent ASP storage usage descriptors
648	288	BINARY(4)	Number of independent ASP storage usage descriptors
652	28C	BINARY(4)	Number of independent ASP storage usage descriptors returned
656	290	BINARY(4)	Length of an independent ASP storage usage descriptor
660	294	CHAR(1)	Local password management
		ARRAY(*) CHAR(10)	Supplemental groups
		CHAR(*)	Home directory
		CHAR(*)	Locale path name
		ARRAY(*) CHAR(*)	Independent ASP storage usage descriptors

Field Descriptions

Accounting code.

The accounting code that is associated with this user. If the user does not have an accounting code, this field is blank.

Assistance level. The user interface that the user will use.

The field contains one of the following values:

*SYSVAL   The system value QASTLVL determines which user interface the user is using.
*BASIC   The Operational Assistant user interface.
*INTERMED   The system user interface.
*ADVANCED   The expert system user interface.

Attention-key-handling program library name. The name of the library where the program is located. This field can contain the special value of *LIBL. If the program name is a special value, this field is blank.

Attention-key-handling program name. The Attention-key-handling program for this user.

This field may contain one of the following special values:

*SYSVAL   The system value QATNPGM determines the user's Attention-key-handling program.
*NONE   No Attention-key-handling program is used.

Bytes available. The number of bytes of data available to be returned to the user. If all data is returned, this is the same as the number of bytes returned. If the receiver variable was not big enough to contain all of the data, this is the number of bytes that can be returned.

Bytes returned. The number of bytes of data returned to the user. This is the lesser of the number of bytes available to be returned or the length of the receiver variable.

Character code set ID. The character code set ID to be used by the system f or this user.

This field can contain the following special value:

-2   The system value QCCSID is used to determine the user's character code set ID.

Character identifier control. The character identifier control for the user.

This field can contain the following special values:

*SYSVAL   The value QCHRIDCTL system value will be used to determine the CHRID control for this user.
*DEVD   The *DEVD special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.
*JOBCCSID   The *JOBCCSID special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

Country or region ID. The country or region ID used by the system for this user.

This field can contain the following special value:

*SYSVAL   The system value QCNTRYID is used to determine the user's country or region ID.

Current library name. This field contains the name of the user's current library. If the user does not have a current library, this field is *CRTDFT.

Date password expires. The date the user's password expires, in *DTS (date-time stamp) format. If the user's password will not expire (password expiration interval of *NOMAX) or the user's password is set to expire, then this field is blank.

Days until password expires. The number of days until the password will expire.

This field contains one of the following values:

0	The password is expired.
1-7	The number of days until the password expires.
-1	The password will not expire in the next 7 days.

Digital certificate indicator. Whether there are digital certificates associated with this user.

Possible values follow:

0	There are no digital certificates associated with this user.
1	There is at least one digital certificates associated with this user.

Display sign-on information. Whether the sign-on information display is shown when the user signs on.

The field contains one of the following values:

SYSVAL	The system value QDSPSGNINF determines if the sign-on information display is shown when the user signs on.
YES	The sign-on information display is shown when the user signs on.
NO	The sign-on information display is not shown when the user signs on.

Group authority. The authority the user's group profile has to objects the user creates.

The field contains one of the following values:

NONE	The group profile has no authority to the objects the user creates. If the user does not have a group profile, the field contains this value.
ALL	The group profile has all authority to the objects the user creates.
CHANGE	The group profile has change authority to the objects the user creates.
USE	The group profile has use authority to the objects the user creates.
EXCLUDE	The group profile has exclude authority to the objects the user creates.

Group authority type. The type of authority the user's group profile has to objects the user creates.

The field contains one of the following values:

PRIVATE	The group profile has a private authority to the objects the user creates. If the user does not have a group profile, the field contains this value.
PGP	The group profile will be the primary group for objects the user creates.

Group ID number. The group ID number for the user profile. The group ID number is used to identify the user when it is a group and a member of the group is using the integrated file system.

Possible values follow:

0	(*NONE) The user does not have a GID.
1-4294967294	A valid GID.

Group member indicator. Whether this user is a group that has members.

Possible values follow:

0	The user is not a group, or is a group but does not have any members.
1	The user is a group that has members.

Group profile name. The name of the group profile. If the user does not have a group profile, this field is *NONE.

Highest scheduling priority. The highest scheduling priority the user is allowed to have for each job submitted to the system. The priority is a value from 0 through 9, with 0 being the highest priority.

Home directory. The home directory for this user profile. The home directory is the user's initial working directory. The working directory, associated with a process, is used in path name resolution in the directory file system for path names that do not begin with a slash (/).

The structure for the home directory name returned is:

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	CCSID of the returned home directory
4	4	CHAR(2)	Country or region ID
6	6	CHAR(3)	Language ID
9	9	CHAR(3)	Reserved
12	C	BINARY(4)	Flags
16	10	BINARY(4)	Number of bytes in the home directory name
20	14	CHAR(2)	Home directory delimiter
22	16	CHAR(10)	Reserved
32	26	CHAR(*)	Home directory name value

Independent ASP storage usage descriptors An array of descriptors that contains the name of the independent ASP, the maximum amount of auxiliary storage (in kilobytes) that can be assigned to store permanent objects owned by the user on the independent ASP, and the amount of auxiliary storage (in kilobytes) occupied by this user's owned objects on the independent ASP. If the user does not have a maximum amount of allowed storage on an independent ASP, the maximum storage field in the descriptor will be set to -1 for *NOMAX.

The structure for the independent ASP storage usage descriptor is:

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Independent ASP name
10	A	CHAR(2)	Reserved

Offset			
Dec	Hex	Type	Field
12	C	BINARY(4)	Maximum allowed storage
16	10	BINARY(4)	Storage used

Initial menu name. The initial menu for the user. This field can contain the special value *SIGNOFF.

Initial menu library name. The name of the library that the initial menu is in. This field can contain the special value of *LIBL. If the menu name is *SIGNOFF, this field is blank.

Initial program name. The initial program for the user. If the user does not have an initial program, this field is *NONE.

Initial program library name. The name of the library that the initial program is in. This field can contain the special value of *LIBL. If the program name is *NONE, this field is blank.

Job description name. The name of the job description used for jobs that start through subsystem work station entries.

Job description library name. The name of the library that the job description is in. This field can contain the special value *LIBL.

Keyboard buffering. This field indicates the keyboard buffering value that is used when a job is initialized for this user.

The field contains one of the following values:

*SYSVAL	The system value QKBDBUF determines the keyboard buffering value for this user.
*YES	The type-ahead and attention-key buffering options are both on.
*NO	The type-ahead and attention-key buffering options are not on.
*TYPEAHEAD	The type-ahead option is on, but the attention-key buffering option is not.

Language ID. The language ID used by the system for this user.

This field can contain the following special value:

*SYSVAL	The system value QLANGID is used to determine the user's language ID.

Length of an independent ASP storage usage descriptor. The length (in bytes) of one independent ASP storage usage descriptor.

Length of home directory. The length (in bytes) of the home directory entry.

Length of locale path name. The length (in bytes) of the locale path name.

Limit capabilities. Whether the user has limited capabilities.

The field contains one of the following values:

*PARTIAL	The user cannot change his initial program or current library.
*YES	The user cannot change his initial menu, initial program, or current library. The user cannot run commands from the command line.
*NO	The user is not limited.

Limit device sessions. Whether the user is limited to one device session.

The field contains one of the following values:

*SYSVAL	The system value QLMTDEVSSN determines if the user is limited to one device session.
*YES	The user is limited to one device session.
*NO	The user is not limited to one device session.

Locale job attributes. The job attributes that are taken from the user's locale path name. If a particular job attribute is taken from the locale path name, the specific field is Y (yes). If not, the specific field is N (no).

The possible values follow:

*NONE   CHAR(1)

No job attributes are used from the locale path name at the time a job is started for this user profile.

*SYSVAL   CHAR(1)

The job attributes assigned from the locale path name are determined by the system value QSETJOBATR at the time a job is started for this user profile.

*CCSID   CHAR(1)

The coded character set identifier is set from the locale path name at the time a job is started for this user profile.

*DATFMT   CHAR(1)

The date format is set from the locale path name at the time a job is started for this user profile.

*DATSEP   CHAR(1)

The date separator is set from the locale path name at the time a job is started for this user profile.

*SRTSEQ   CHAR(1)

The sort sequence is set from the locale path name at the time a job is started for this user profile.

*TIMSEP   CHAR(1)

The time separator is set from the locale path name at the time a job is started for this user profile.

*DECFMT   CHAR(1)

The decimal format is set from the locale path name at the time a job is started for this user profile.

Reserved   CHAR(8)

An ignored field.

Locale path name. The locale path name that is assigned to the user profile when a job is started. This field can contain a special value or a locale path name. If a special value is returned, the length of the value is 10 and the value returned is one of the following:

*C	The C locale path name is assigned.
*NONE	No locale path name is assigned.
*POSIX	The POSIX locale path name is assigned.
*SYSVAL	The QLOCALE system value is used to determine the locale path name.

If the value returned in this field is not a special value, it is returned in the following format.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	CCSID
4	4	CHAR(2)	Country or region ID
6	6	CHAR(3)	Language ID
9	9	CHAR(3)	Reserved
12	C	BINARY(4)	Flags
16	10	BINARY(4)	Locale path name length
20	14	CHAR(2)	Locale path name delimiter character
22	16	CHAR(10)	Reserved
32	26	CHAR(*)	Locale path name

Local password management. If password is managed locally, this field contains a Y. If not, this field is N.

Maximum allowed storage. The maximum amount of auxiliary storage (in kilobytes) that can be assigned to store permanent objects owned by the user. If the user does not have a maximum amount of allowed storage, this field contains -1 for *NOMAX.

Message queue name. The name of the message queue that is used by this user.

Message queue library name. The name of the library the message queue is in. This field can contain the special value *LIBL.

Message queue delivery method. How the messages are delivered to the message queue used by the user. This field contains one of the following special values:

*BREAK	The job to which the message queue is assigned is interrupted when a message arrives on the message queue.
*DFT	Messages requiring replies are answered with their default reply.
*HOLD	The messages are held in the message queue until they are requested by the user or program.
*NOTIFY	The job to which the message queue is assigned is notified when a message arrives on the message queue.

Message queue severity. The lowest severity that a message can have and still be delivered to a user in break or notify mode. The severity is a value from 0 through 99.

No password indicator. If *NONE is specified for the password in the user profile, this field contains a Y. If not, this field is N.

Number of independent ASP storage usage descriptors. The total number of independent ASP storage usage descriptors. The number of independent ASP storage usage descriptors will be 0 if the user does not own any objects on any independent ASPs.

Number of independent ASP storage usage descriptors returned. The number of independent ASP storage usage descriptors returned in the array. The number of independent ASP storage usage descriptors will be 0 if the user does not own any objects on any independent ASPs.

Number of supplemental groups. The number of supplemental groups. The number of supplemental groups will be zero if the user does not have any supplemental groups.

Object auditing value. The current user's object auditing value.

≫ **Note:** The user must have either *ALLOBJ or *AUDIT special authority to retrieve the object auditing value. ≪

The field contains one of the following values:

*NONE	No additional object auditing is done for the current user.
*CHANGE	Object changes are audited for the current user if the object's auditing value is *USRPRF.
*ALL	Object read and change operations are audited for the current user if the object's auditing value is *USRPRF.
≫ *NOTAVL	The user is not allowed to retrieve the object auditing value. ≪

Offset to array of independent ASP storage usage descriptors. The offset from the beginning of the receiver variable to the start of the array of independent ASP storage usage descriptors.

Offset to array of supplemental groups. The offset from the beginning of the receiver variable to the start of the array of supplemental groups.

Offset to home directory. The offset from the beginning of the receiver variable to the start of the home directory entry.

Offset to locale path name. The offset from the beginning of the receiver variable to the start of the locale path name.

Output queue name. The output queue used by this user.

This field can contain one of the following special values:

*WRKSTN	The output queue assigned to the user's work station is used.
*DEV	An output queue with the same name as the device specified in the printer device parameter is used by the user.

Output queue library name. The name of the library where the output queue is located. This field can contain the special value *LIBL. If the output queue is *WRKSTN or *DEV, this field is blank.

Owner. This field indicates who is to own objects created by this user.

The field contains one of the following values:

*USRPRF	The user owns any objects the user creates. If the user does not have a group profile, the field contains this value.
*GRPPRF	The user's group profile owns any objects the user creates.

Password change date. The date the user's password was last changed, in *DTS (date-time stamp) format.

Password expiration interval. The number of days (from 1 through 366) the user's password can remain active before it must be changed.

This field may contain one of the following special values:

0	The system value QPWDEXPITV is used to determine the user's password expiration interval.
-1	The user's password does not expire (*NOMAX).

Previous sign-on date and time. The date and time the user last signed on. The 13 characters are:
- CHAR(1): Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
- CHAR(6): Date, in YYMMDD (year, month, day) format.
- CHAR(6): Time, in HHMMSS (hours, minutes, seconds) format.

If the user has never signed on the system, this field is blank.

Print device. The printer used to print for this user.

This field can contain one of the following special values:

WRKSTN	The printer assigned to the user's work station is used.
SYSVAL	The default system printer specified in the system value QPRTDEV is used.

Reserved. An ignored field.

Set password to expire. Whether the user's password is set to expire, requiring the user to change the password when signing on.

This field contains one of the following values:

Y	The user's password is set to expire.
N	The user's password is not set to expire.

Sign-on attempts not valid. The number of sign-on attempts that were not valid since the last successful sign-on.

Sort sequence table name. The name of the sort sequence table used for string comparisons. The following possible special values can also be returned:

HEX	The hexadecimal values of the characters are used to determine the sort sequence.
LANGIDUNQ	A unique-weight sort table associated with the language specified.
LANGIDSHR	A shared-weight sort table associated with the language specified.
SYSVAL	The system value QSRTSEQ.

Sort sequence table library name. The name of the library that is used to locate the sort sequence table. This information is blank if the program does not contain any sort sequence information.

Special authorities. The special authorities the user has. If the user has the special authority, the field is Y. If not, the field is N.

This field contains the following fields:

ALLOBJ	CHAR(1)
	All object. Whether the user has all object special authority.
SECADM	CHAR(1)
	Security administrator. Whether the user has security administrator special authority.
JOBCTL	CHAR(1)
	Job control. Whether the user has job control special authority.
SPLCTL	CHAR(1)
	Spool control. Whether the user has spool control special authority.

*SAVSYS	CHAR(1)
	Save system. Whether the user has save system special authority.
*SERVICE	CHAR(1)
	Service. Whether the user has service special authority.
*AUDIT	CHAR(1)
	Audit. Whether the user has audit special authority.
*IOSYSCFG	CHAR(1)
	Input/output system configuration. Whether the user has input/output system configuration special authority.
Reserved	CHAR(7)
	An ignored field.

Special environment. The special environment the user operates in after signing on.

This field contains one of the following special values:

*SYSVAL	The system value QSPCENV is used to determine the user's special environment.
*NONE	The user operates in the i5/OS environment.
*S36	The user operates in the System/36 environment.

Status. The status of the user profile.

This field contains one of the following values:

*ENABLED	The user profile is enabled; therefore, the user is able to sign on.
*DISABLED	The user profile is disabled; therefore, the user cannot sign on.

Storage used. The amount of auxiliary storage (in kilobytes) occupied by this user's owned objects.

Supplemental groups. The array of supplemental groups for the user profile.

Text description. The descriptive text for the user profile.

User action audit level. The action audit values for this user. If the user has a specific audit value, the field is Y. If not, the field is N. ≫ If the user is not allowed to retrieve the action audit level, the field is blank.

Note: The user must have either *ALLOBJ or *AUDIT special authority to retrieve the user action audit level. ≪

This field contains the following:

*CMD	CHAR(1)
	The user has the *CMD audit value specified in the user profile.
*CREATE	CHAR(1)
	The user has the *CREATE audit value specified in the user profile.
*DELETE	CHAR(1)
	The user has the *DELETE audit value specified in the user profile.

*JOBDTA   CHAR(1)

The user has the *JOBDTA audit value specified in the user profile.

*OBJMGT   CHAR(1)

The user has the *OBJMGT audit value specified in the user profile.

*OFCSRV   CHAR(1)

The user has the *OFCSRV audit value specified in the user profile.

*OPTICAL   CHAR(1)

The user has the *OPTICAL audit value specified in the user profile.

*PGMADP   CHAR(1)

The user has the *PGMADP audit value specified in the user profile.

*SAVRST   CHAR(1)

The user has the *SAVRST audit value specified in the user profile.

*SECURITY   CHAR(1)

The user has the *SECURITY audit value specified in the user profile.

*SERVICE   CHAR(1)

The user has the *SERVICE audit value specified in the user profile.

*SPLFDTA   CHAR(1)

The user has the *SPLFDTA audit value specified in the user profile.

*SYSMGT   CHAR(1)

The user has the *SYSMGT audit value specified in the user profile.

Reserved   CHAR(51)

An ignored field.

User class name.

This field contains one of the following special values:

*SECOFR   The user has a class of security officer.
*SECADM   The user has a class of security administrator.
*PGMR   The user has a class of programmer.
*SYSOPR   The user has a class of system operator.
*USER   The user has a class of end user.

User ID number. The user ID (UID) number for the user profile. The UID is used to identify the user when it is using the integrated file system.

User options. The options for users to customize their environment. This field contains the following fields:
- CHAR(1): Show keywords (*CLKWD). Whether the keywords are shown when a CL command is displayed. If the keywords are to be shown, this field is Y. If not, this field is N.
- CHAR(1): Show detailed information (*EXPERT). Whether more detailed information is shown when the user is defining or changing the system using edit or display object authority. This user option is independent of the ASTLVL parameter on the user profile and the ASTLVL parameter available on commands. If the details are to be shown, this field is Y. If not, this field is N.
- CHAR(1): Full screen help (*HLPFULL). Whether UIM online help is to be displayed on a full screen or in a window. If the full screen is to be shown, this field is Y. If not, this field is N.

- CHAR(1): Show status message (*STSMSG). Whether status messages sent to the user are shown. If the status messages are to be shown, this field is Y. If not, this field is N.
- CHAR(1): Do not show status message (*NOSTSMSG). Whether status messages sent to the user are not shown.
- CHAR(1): Roll key direction change (*ROLLKEY). Whether the opposite action from the system default for roll keys is taken or not. If the opposite action is to be taken, this field is Y. If not, this field is N.
- CHAR(1): Printing complete message (*PRTMSG). Whether a message is sent to the user when a spooled file is printed or not. If a message is to sent to the user, this field is Y. If not, this field is N.
- CHAR(29): Reserved.

User profile name. The name of the user profile for which the information is returned.

Error Messages

Message ID	Error Message Text
CPF2203 E	User profile &1 not correct.
CPF2225 E	Not able to allocate internal system object.
CPF3CF1 E	Error code parameter not valid.
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.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Retrieve Users Authorized to an Object (QSYRTVUA) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Returned records feedback information	Output	Char(*)
4	Length of returned records feedback information	Input	Binary(4)
5	Format name	Input	Char(8)
6	Object name	Input	Char(*)
7	Length of object name	Input	Binary(4)
8	Error code	I/O	Char(*)

Optional Parameter Group:

9	Symbolic link	Input	Char(10)

Default Public Authority: *USE
Threadsafe: No

The Retrieve Users Authorized to an Object (QSYRTVUA) API provides information about the users who are authorized to an object. The API returns the following information:

- A list of users who have a private authority to the object and the authority that the users have
- The public authority for the object

- Other authority information for the object, such as the name of the owner, the primary group, and the authorization list securing the object
- For objects in the QDLS file system, the sensitivity level of the object

This API provides information that is similar to the Display Authority (DSPAUT) command.

Authorities and Locks

*X is required for all directories in the path.

Authority to Object
> *OBJMGT

*Authority to Object (QSYS.LIB *AUTL object)*
> No authority is required

Authority to Object (QDLS file system)
> *ALL

Authority to Object (QOPT file system)
> *USE

Required Parameter Group

Receiver variable
> OUTPUT; CHAR(*)

> The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable that is specified in the user program. If the length of receiver variable parameter that is specified is larger than the allocated size of the receiver variable that is specified in the user program, the results are not predictable.

Returned records feedback information
> OUTPUT; CHAR(*)

> Information about the object and information about the entries that are returned in the receiver variable.

> See "Format of Returned Records Feedback Information" on page 150 for details.

Length of returned records feedback information
> INPUT; BINARY(4)

> The length of the returned records feedback information provided. The length of the returned records feedback information parameter may be specified up to the size of the returned records feedback information variable specified in the user program. If the length of the returned records feedback information parameter specified is larger than the allocated size of the returned records feedback information variable that is specified in the user program, the results are not predictable. The minimum length is 16 bytes.

Format name
> INPUT; CHAR(8)

> The name of the format that is used to return information about the users who are authorized to the object.

You can specify this format:

"RTUA0100 Format"   Each entry contains the name of the profile that is authorized to the object, whether the profile is a user profile or a group profile, and the profile's authority to the object.

Object name
    INPUT; CHAR(*)

    The object name.

    If the length of the object name is greater than 0, then this parameter is assumed to be a path name represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

    If the length of the object name is -1, then this parameter is assumed to be a Qlg_Path_Name_T structure that contains a path name or a pointer to a path name. For more information on the Qlg_Path_Name_T structure, see Path name format.

Length of object name
    INPUT; BINARY(4)

    The length of the object name. If the length is -1, the object name parameter is assumed to be a Qlg_Path_Name_T structure.

Error code
    I/O; CHAR(*)

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

Optional Parameter Group

Symbolic link
    INPUT; CHAR(10)

    If the last component in the path name is a symbolic link, specifies whether or not to retrieve users authorized to the symbolic link or users authorized to the object pointed to by the symbolic link.

    The valid values are:

*NO   The users authorized to the symbolic link object are not retrieved. The users authorized to the object pointed to by the symbolic link are retrieved.

    This is the default value when the Symbolic link parameter is not specified

*YES   If the object is a symbolic link, the users authorized to the symbolic link object are retrieved. The users authorized to the object pointed to by the symbolic link are not retrieved.

Receiver Variable Description

The following table describes the order and format of the data that is returned in the receiver variable for each user that is authorized to the object. For detailed descriptions of the fields in the table, see "Field Descriptions" on page 150.

RTUA0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Profile name

Offset			
Dec	Hex	Type	Field
10	0A	CHAR(1)	User or group indicator
11	0B	CHAR(10)	Data authority
21	15	CHAR(1)	Authorization list management
22	16	CHAR(1)	Object management
23	17	CHAR(1)	Object existence
24	18	CHAR(1)	Object alter
25	19	CHAR(1)	Object reference
26	1A	CHAR(10)	Reserved
36	24	CHAR(1)	Object operational
37	25	CHAR(1)	Data read
38	26	CHAR(1)	Data add
39	27	CHAR(1)	Data update
40	28	CHAR(1)	Data delete
41	29	CHAR(1)	Data execute
42	2A	CHAR(10)	Reserved

Format of Returned Records Feedback Information

For a description of the fields in this format, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned in the returned records feedback information
4	4	BINARY(4)	Bytes available in the returned records feedback information
8	8	BINARY(4)	Bytes returned in the receiver variable
12	C	BINARY(4)	Bytes available in the receiver variable
16	10	BINARY(4)	Number of authorized users
20	14	BINARY(4)	Entry length for each authorized user returned
24	18	CHAR(10)	Owner
34	22	CHAR(10)	Primary group
44	2C	CHAR(10)	Authorization list
54	36	CHAR(1)	Sensitivity level

Field Descriptions

Authorization list. The name of the authorization list that is securing the object. If there is no authorization list that secures the object, this field is *NONE.

Authorization list management. Whether the user has this authority to the object. This field is only valid if the object is an authorization list.

This field contains one of the following values:

0	The user does not have this authority.
1	The user has this authority.

Bytes available in the receiver variable. The number of bytes of data that is available to be returned to the user in the receiver variable. All available data is returned if enough space is provided.

Bytes available in the returned records feedback information. The number of bytes of data available to be returned to the user in the returned records feedback information. All available data is returned if enough space is provided.

Bytes returned in the receiver variable. The number of bytes of data that is returned to the user in the receiver variable.

Bytes returned in the returned records feedback information. The number of bytes of data returned to the user in the returned records feedback information.

Data add. Whether the user has this authority to the object.

This field contains one of the following values:

0	The user does not have this authority.
1	The user has this authority.

Data authority. The data authority that the authorized user has to the object.

This field contains one of the following values:

*RWX	The user has object operational, read, add, update, delete, and execute authorities to the object.
*RW	The user has object operational, read, add, update, and delete authorities to the object.
*RX	The user has object operational, read, and execute authorities to the object.
*WX	The user has object operational, add, update, delete, and execute authorities to the object.
*R	The user has object operational and read authorities to the object.
*W	The user has object operational, add, update, and delete authorities to the object.
*X	The user has object operational and execute authorities to the object.
*EXCLUDE	The user has no authority to the object.
*AUTL	The public authority to the object comes from the public authority on the authorization list that secures the object. This value can be returned only if there is an authorization list that secures the object and the authorized user is *PUBLIC.
USER DEF	The user has some combination of data rights that do not relate to a special value. The API user should check the individual authorities for the user to determine what authority the user has to the object.

Data delete. Whether the user has this authority to the object.

This field contains one of the following values:

0	The user does not have this authority.
1	The user has this authority.

Data execute. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Data read. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Data update. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Entry length for each authorized user returned. The entry length, in bytes, of each entry in the list of users who are authorized to the object.

Number of authorized users. The number of complete entries in the list of users who are authorized to the object. A value of zero is returned if the list is empty.

Object alter. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Object existence. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Object management. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Object operational. Whether the user has this authority to the object.

This field contains one of the following values:

0   The user does not have this authority.
1   The user has this authority.

Object reference. Whether the user has this authority to the object.

This field contains one of the following values:

0	The user does not have this authority.
1	The user has this authority.

Owner. The name of the owner of the object. If the owner has no authority, no authorized user entry is returned for the owner.

This field can contain the following special value:

*NOUSRPRF   The user profile that owns this object does not exist on this system.

Primary group. The name of the primary group for the object. If the primary group has no authority, no authorized user entry is returned for the primary group.

This field can contain the following special value:

*NONE	There is no primary group for the object.
*NOUSRPRF	The user profile that is the primary group for this object does not exist on this system.

Profile name. The name of the user profile that is authorized to the object.

This field can contain the following special values:

*PUBLIC	Public authority (the authority used by users who are not privately authorized) to the object. This is the first entry that is returned.
*NOUSRPRF	The user profile that is authorized to this object does not exist on this system.
*NTWIRF	The NetWare inherited rights filter to the object (only valid for the QNetWare file system).
*NTWEFF	The NetWare effective rights to the object (only valid for the QNetWare file system).

Reserved. An ignored field.

Sensitivity level. The sensitivity level of a QDLS object. For all other objects, this field contains 0.

This field contains one of the following values:

0	This value does not apply to this object.
1	(None) The object has no sensitivity restrictions.
2	(Personal) The object contains information intended for the user as an individual.
3	(Private) The object contains information that should be accessed only by the owner.
4	(Confidential) The object contains information that should be handled according to company procedures.

User or group indicator. Whether this user is a user profile or a group profile.

This field contains one of the following values:

0	This user is not a user or a group. This value is returned for special values such as *PUBLIC.
1	This user is a user profile.
2	This user is a group profile.

Error Messages

Message ID	Error Message Text
CPFA0A9 E	Object not found. Object is &1.
CPFA0CE E	Error occurred with path name parameter specified.
CPFA09C E	Not authorized to object. Object is &1.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Set Encrypted User Password (QSYSUPWD) API

Required Parameter Group:

1	Receiver variable from QSYRUPWD	Input	Char(*)
2	Format	Input	Char(8)
3	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Set Encrypted User Password (QSYSUPWD) API sets the encrypted password data for the specified user profile by using the receiver variable that was retrieved with the Retrieve Encrypted User Password (QSYRUPWD) API.

The QSYSUPWD API changes the following fields in addition to the password values:
- The password expiration field is set to *NO.
- The password change date field is updated.
- The user profile change date is updated.

Note: If an error occurs while attempting to set the iSeries Support for Windows Network Neighborhood (iSeries NetServer) password, the CPF22F0 error is returned and no other encrypted password data is set.

If the local password management (LCLPWDMGT) value for the specified user profile is *NO, then the local i5/OS password will be set to *NONE. Also, if the LCLPWDMGT value was *NO for the user profile on the system where the QSYRUPWD API was called, then the local i5/OS password will be set to *NONE.

The QSYRUPWD (Retrieve Encrypted User Password) API does not retrieve product-level encrypted data that may be associated with a user profile. Additional steps may be needed after the QSYSUPWD (Set Encrypted User Password) API is run, to ensure that product-level encrypted data is correct.

Authorities and Locks

User Profile Authority
        *ALLOBJ and *SECADM

Required Parameter Group

Receiver variable from QSYRUPWD
        INPUT; CHAR(*)

The variable that is used to set the encrypted password for the user. The receiver variable from the QSYRUPWD API must be used as input to this API. For this API to successfully set the encrypted password for the user, the bytes returned value must be equal to the bytes available value in the input data. The input data must be retrieved from the receiver variable used by the QSYRUPWD API.

Format
        INPUT; CHAR(8)

The name of the format that is used to set the user's encrypted password data.

The following values are allowed:

"UPWD0100 Format"   Encrypted password will be set

Error code
        I/O; CHAR(*)

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

UPWD0100 Format

The following table describes the input variable that is to be passed as the first parameter to QSYSUPWD. This input variable must be the same data as the receiver variable that is returned by the QSYRUPWD API. The receiver variable, returned by the QSYRUPWD API, cannot be changed in any way prior to passing the data as input to the QSYSUPWD API. If this data is changed, the QSYSUPWD API will not be able to successfully change the password for the user. For detailed descriptions of the fields in the tables, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User profile name
18	12	CHAR(*)	Encrypted user password data

Field Descriptions

Bytes available. The number of bytes of data available when retrieved by the QSYRUPWD API. For the QSYSUPWD API to successfully set the encrypted password for the user, this value must be equal to the bytes returned value. If the bytes available field is greater than the bytes returned field, this input cannot be used to successfully set the encrypted password for the user.

Bytes returned. The number of bytes of data.

Encrypted user password data. The encrypted password data for the user profile.

User profile name. The name of the user profile for which the password will be changed.

Error Messages

Message ID	Error Message Text
CPD2201 E	System user profile cannot be changed.
CPF2203 E	User profile &1 not correct.
CPF2225 E	Not able to allocate internal system object.
CPF222E E	&1 special authority is required.
CPF22F0 E	Unexpected errors occurred during processing.
CPF3C21 E	Format name &1 is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF4AB2 E	Receiver variable from QSYRUPWD has been altered.
CPF9801 E	Object &2 in library &3 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R7

Set Job User Identity (QWTSJUID) API

Required Parameter:

1	Operation requested	Input	Binary(4)
2	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: Conditional; see "Usage Notes" on page 157.

The Set Job User Identity (QWTSJUID) API has two operations that can be used to explicitly set the job user identity of the current job. The two operations are set and clear. The set operation explicitly sets the job user identity to the name of the current user profile of the thread in which the API is called. The clear operation clears any job user identity that was previously set by the QWTSJUID API or the QwtSetJuid() function, and the default job user identity will then take effect.

The **job user identity** is the name of the user profile by which this job is known to other jobs. The job user identity is used for authorization checks when other jobs on the system attempt to operate against this job. Examples of functions that operate against another job include the Start Service Job (STRSRVJOB) command, the Retrieve Job Information (QUSRJOBI) API, the Change Job (QWTCHGJB) API, all job control commands, and functions that send signals from one job to another.

The job user identity is not used to make authorization checks from within this job. Authorization to perform a function is always based on the current user profile of the thread in which the function is called.

This API is intended to be used by either multithreaded servers or single threaded servers that want to establish a job user identity that remains constant, regardless of the user profile under which it processes individual client requests.

When a job user identity has not been explicitly set by an API, a job running single threaded will have a job user identity that is the same as the current user profile that the job is running under at the time. When a job user identity has not been explicitly set by an API, a job running multithreaded will have a job user identity that is the name of the user profile that the job was running under at the time it became multithreaded.

Authorities and Locks

If the job user identity is currently set, then either *USE authority to the user profile associated with the job user identity or all object (*ALLOBJ) special authority is required. If the job user identity is not already set, then no authorization is required.

Required Parameter

Operation requested
>   INPUT; BINARY(4)

>   The operation to be performed.

>   The possible operations are:

1   Set job user identity to the name of the job's current user profile.
2   Clear the job user identity. The default job user identity will now take effect.

Error code
>   I/O; CHAR(*)

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

Usage Notes

Thread safety considerations:

* The set function is threadsafe.
* The clear function may be called in a job that allows multiple threads, but only while it is running single threaded. The clear function will not be allowed if any secondary threads are active.

Error Messages

Message ID	Error Message Text
CPF180B E	Function &1 not allowed.
CPF2217 E	Not authorized to user profile &1.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V4R3

QwtSetJuid()—Set Job User Identity

Syntax

```
#include <qwtjuid.h>

int QwtSetJuid(void);
```

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

The**QwtSetJuid()** function sets the job user identity of the current job to the name of the current user profile of the thread in which the function is called.

The job user identity is the name of the user profile by which this job is known to other jobs. The job user identity is used for authorization checks when other jobs on the system attempt to operate against this job. Examples of functions that operate against another job include the Start Service Job (STRSRVJOB) command, the Retrieve Job Information (QUSRJOBI) API, the Change Job (QWTCHGJB) API, all job control commands, and functions that send signals from one job to another.

The job user identity is not used to make authorization checks from within this job. Authorization to perform a function is always based on the current user profile of the thread in which the function is called.

This API is intended to be used by either multithreaded servers or single threaded servers that want to establish a job user identity that remains constant, regardless of the user profile under which it processes individual client requests.

When a job user identity has not been explicitly set by an API, a job running single threaded will have a job user identity that is the same as the current user profile that the job is running under at the time. When a job user identity has not been explicitly set by an API, a job running multithreaded will have a job user identity that is the name of the user profile that the job was running under at the time it became multithreaded.

Parameters

None

Authorities and Locks

If the job user identity is currently set, then either *USE authority to the user profile associated with the job user identity or all object (*ALLOBJ) special authority is required. If the job user identity is not already set, then no authorization is required.

Return Value

[EPERM]

> Operation not permitted.

> You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.

API introduced: V4R3

Set Profile Handle (QWTSETP, QsySetToProfileHandle) API

Required Parameter for QWTSETP:

1	Profile handle	Input	Char(12)

Optional Parameter for QWTSETP:

2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Conditional; see "Considerations for Scope and Thread Safety" on page 160.

Syntax for QsySetToProfileHandle:

```
#include <qsyphandle.h>

void QsySetToProfileHandle
        (unsigned char  *Profile_handle,
         void           *Error_code);
```

Service Program: QSYPHANDLE
Default Public Authority: *USE
Threadsafe: Conditional; see "Considerations for Scope and Thread Safety" on page 160.

The Set Profile Handle (OPM, QWTSETP; ILE, QsySetToProfileHandle) API validates the profile handle, locks the user profile, and changes the current thread to run under the user and group profiles represented by the profile handle. Once the change has been made, any open files and objects allocated by the original profile are accessible to the new profile.

No other attributes associated with the user or group profile are replaced. The qualified job name does not change to reflect the new user profile. However, any object created by the thread while running under the new profile is owned by the new profile or its group profile. If the job is running single threaded and the job user identity has not been explicitly set by an API, the job user identity is changed to the name of the new profile. If the job is running multithreaded, the job user identity does not change.

If the profile handle is not valid, the Set Profile Handle API, adds an exception to the job log, and enters a security violation in the QAUDJRN audit journal.

If you use this API to begin running under a specific profile, any spooled files created are, by default, owned by that profile. This is controlled by the spool file owner (SPLFOWN) parameter on the CRTPRTF command and is done by putting the file under a QPRTJOB job. Any spooled file command that references the spooled file with the job special value * will only access those files that were created before the profiles were swapped.

A QPRTJOB job is the name of a job that files are spooled under when the current job's user name is not the same as the user profile currently running. For example, if you use Set Profile Handle to set the profile to user JOE and create a spooled file, the file is spooled under job nnnnnn/JOE/QPRTJOB. This ensures that user JOE owns the spooled file and if that user uses the WRKSPLF command, the file is displayed.

Output Queue Considerations

The output queue a spooled file is placed in may be different after using this API. If the application using this API produces spooled output that needs to be on a secure output queue or the application is expecting the spooled output to be found on a particular output queue, configuration changes may be required. See the Printing topic for information about which output queue contains the spooled output.

Authorities and Locks

None.

Required Parameter

Profile handle
  INPUT; CHAR(12)

  The profile handle returned by the QSYGETPH API or QsyGetProfileHandle API for the user profile to switch the thread to.

Optional Parameter

Error code
> I/O; CHAR(*)

> The structure in which to return error information. For the format of the structure, see Error Code Parameter. This parameter is optional for the QWTSETP API and is omissable for the QsySetToProfileHandle API.

Considerations for Scope and Thread Safety

This API sets the user profile for the thread in which it is called. Thus if the API is called while running multithreaded, it will result in different threads in the same process simultaneously running under different user profiles.

While this API itself is threadsafe, it should only be used in a job that is running multithreaded when all code running in the job is known to be trusted and operating in a coordinated manner. Some considerations when running multiple threads under different user profiles are:

- The design of threads is for every thread in the job to share the same resources. With threads, programs share the same static and heap storage, and by passing pointers, they can get at each other's automatic storage. They also share open files and other resources, such as the same QTEMP library and the profile handles used by the Set Profile Handle API.

- Assume two users are allowed to run their own commands or programs in different threads of a single job. One of the users may be able to read or write data of the other user. This access may occur without the system doing an authority check or even auditing the fact that they read or modified that data.

Error Messages

Message ID	Error Message Text
CPF22AD E	Group profile for user not found.
CPF22E7 E	Profile handle is not valid.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
CPF2225 E	Not able to allocate internal system object.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Set To Profile Token (QSYSETPT, QsySetToPrfTkn) API

»

Required Parameter Group for QSYSETPT:

| 1 | Profile token | Input | Char(32) |
| 2 | Error code | I/O | Char(*) |

Default Public Authority: *USE
Threadsafe: Yes

Syntax for QsySetToPrfTkn:

```
#include <qsyptkn.h>

void QsySetToPrfTkn
     (unsigned char  *Profile_token,
      void           *Error_code);
```

Service Program: QSYPTKN
Default Public Authority: *USE
Threadsafe: Yes

≪

The Set To Profile Token (OPM, QSYSETPT; ILE, QsySetToPrfTkn) API validates the profile token and changes the current thread to run under the user and group profiles represented by the profile token.

The qualified job name does not change to reflect the new user profile. Any object, however, created by the thread while running under the new profile is owned by the new profile or its group profile. If the job is running single threaded and the job user identity has not been explicitly set by an API, the job user identity is changed to the name of the new profile. If the job is running multithreaded, the job user identity does not change.

If the profile token is not valid, this API signals the message CPF2274 and puts an AF-W audit entry in the QAUDJRN audit journal.

If you use this API to begin running under a specific profile, any spooled files created are, by default, owned by that profile. This is controlled by the spool file owner (SPLFOWN) parameter on the CRTPRTF command and is done by putting the spooled file under a QPRTJOB job. Any spooled file command that references the spooled file with the job special value * will access only those files that were created before the profiles were swapped.

QPRTJOB

A QPRTJOB job is the name of a job under which files are spooled when the current job's user name is not the same as the user profile running currently. For example, if you use this API to set the profile to user JOE and create a spooled file, the file is spooled under job nnnnnn/JOE/QPRTJOB. This ensures that user JOE owns the spooled file and if that user uses the WRKSPLF command, the file is displayed.

Output Queue Considerations

The output queue that a spooled file is placed in may be different after using this API. If the application using this API produces spooled output that needs to be on a secure output queue or the application is expecting the spooled output to be found on a particular output queue, configuration changes may be required. See the Printing topic for information about which output queue contains the spooled output.

Authorities and Locks

API Public Authority
     *USE

Required Parameter Group

Profile token
     INPUT; CHAR(32)

     The profile token returned by the Generate Profile Token (QSYGENPT, QsyGenPrfTkn)API or Generate Profile Token From Profile Token (QSYGENFT, QsyGenPrfTknFromPrfTkn) API that represents the user profile to which to switch.

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
CPF18A8 E	Error occured during set profile to profile token.
CPF2225 E	Not able to allocate internal system object.
CPF2274 E	Profile token is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

Usage Notes

Considerations for Scope and Thread Safety

This API sets the user profile for the thread in which it is called. Thus, if the API is called while running multithreaded, it will result in different threads in the same process simultaneously running under different user profiles.

While this API itself is threadsafe, it should only be used in a job that is running multithreaded when all code running in the job is known to be trusted and operating in a coordinated manner. Some considerations when running multiple threads under different user profiles are:

- The design of threads is for every thread in the job to share the same resources. With threads, programs share the same static and heap storage, and by passing pointers, they can get at each other's automatic storage. They also share open files and other resources, such as the same QTEMP library and the profile tokens used by this API.

- Assume two users are allowed to run their own commands or programs in different threads of a single job. One of the users may be able to read or write data of the other user. This access may occur without the system doing an authority check or even auditing the fact that they read or modified the data.

API introduced: V4R5

Sign Buffer (QYDOSGNB, QydoSignBuffer)

> Required Parameter Group:

1	Buffer to sign	Input	Char(*)
2	Description of buffer to sign	Input	Char(*)
3	Number of descriptions of buffer to sign	Input	Binary(4)
4	Application identifier	Input	Char(*)
5	Length of application identifier	Input	Binary(4)
6	Resulting signature	Output	Char(*)
7	Length of resulting signature provided	Input	Binary(4)
8	Format of resulting signature	Input	Char(8)
9	Error Code	I/O	Char(*)

Service Program Name: QYDOBUFFER
Default Public Authority: *USE
Threadsafe: No

The Sign Buffer (OPM, QYDOSGNB; ILE, QydoSignBuffer) API allows the local system to certify that the series of bytes being signed is trustworthy. It does this by generating a digital signature for those bytes and returning this signature to the caller.

The application identifier will be used to find the certificate needed to sign this object. The certificate will be used later to verify the contents of this object have not changed and this certificate will be reported as having signed this object.

Authorities and Locks

API Public Authority
>*USE.

Authority Required
>To use this API, you must be authorized to the object signing applications function associated with your application identifier through iSeries Navigator's application administration support. The Change Function Usage Information(QSYCHFUI) API, with a function ID of the same name as the application identifier, also can be used to change the list of users that are allowed to use this application identifier.

Required Parameter Group

Buffer to sign
>INPUT; CHAR(*)
>
>The buffer of data to be signed. Only the part of the object described in the Description of buffer to sign will be signed.

Description of buffer to sign
>INPUT; CHAR(*)
>
>Array of offsets and lengths to the data to be signed. The API will treat these bytes as if they were a contiguous stream of bytes. The offset is from the start of the buffer.
>
>The format of the description of the data to sign is in the following table. For detailed descriptions of the fields in this table, see "Field Descriptions" on page 164.

Offset			
Dec	Hex	Type	Field
0	0	Binary(4)	Offset to start of first series of bytes to sign
4	4	Binary(4)	Length of first series of bytes to sign
n	n	Binary(4)	Offset to start of next series of bytes to sign
n+4	n+10	Binary(4)	Length of next series of bytes to sign

Number of descriptions of buffer to sign
>INPUT; BINARY(4)
>
>Number of offsets and lengths needed to describe what parts of the buffer should be signed.

Application identifier
>INPUT; CHAR(*)
>
>The user-supplied application ID to sign objects with. The application type must be 4 (object signing) and it must be assigned to a valid certificate label.

Length of application identifier
      INPUT; BINARY(4)

      The length of the specified application identifier. This length must be a value from 1 to 30.

Resulting signature
      OUTPUT; CHAR(*)

      Area to contain the signature to be returned by the API. See "Resulting signature formats" for details on the format of this parameter. This field may be NULL if the length of resulting signature provided is 0.

Length of resulting signature provided
      INPUT; BINARY(4)

      The length of the area provided to contain the returned signature.

Format of resulting signature
      INPUT; CHAR(8)

      The format of the results of the signing operation.

"SGNB0100 format" on page 165	Just the signature itself is returned. The signature will be in PKCS #1 block type 01 format.
"SGNB0200 format" on page 165	The signature itself and the certificate label needed to verify the signature are returned. The signature will be in PKCS #1 block type 01 format.
"SGNB0300 format" on page 165	The signature itself and the ASN.1 encoded certificate itself needed to verify the signature are returned. The signature will be in PKCS #1 block type 01 format.
"SGNB0400 format" on page 165	The signature itself and the distinguished name of the certificate needed to verify the signature are returned. The signature will be in PKCS #1 block type 01 format.

Error code
      I/O; CHAR(*)

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

Field Descriptions

Length of first series of bytes to sign The number of bytes, including the first byte in the series, to be included in the signature.

Length of next series of bytes to sign The number of bytes, including the first byte in the series, to be included in the signature.

Offset to start of first series of bytes to sign. An offset to the first byte of a series of 1 or more bytes of data to be included in the signature.

Offset to start of next series of bytes to sign. An offset to the first byte of a series of 1 or more bytes of data to be included in the signature.

Resulting signature formats

For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 166.

SGNB0100 format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset to start of signature
4	4	BINARY(4)	Length of signature
		CHAR(*)	Signature

SGNB0200 format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset to start of signature
4	4	BINARY(4)	Length of signature
8	8	BINARY(4)	Offset to start of certificate label
12	0C	BINARY(4)	Length of certificate label
		CHAR(*)	Signature
		CHAR(*)	Certificate label

SGNB0300 format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset to start of signature
4	4	BINARY(4)	Length of signature
8	8	BINARY(4)	Offset to start of certificate
12	0C	BINARY(4)	Length of certificate
		CHAR(*)	Signature
		CHAR(*)	Certificate

SGNB0400 format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Offset to start of signature
4	4	BINARY(4)	Length of signature
8	8	BINARY(4)	Offset to start of distinguished name
12	0C	BINARY(4)	Length of distinguished name
		CHAR(*)	Signature
		CHAR(*)	Distinguished name

Field Descriptions

Certificate. The ASN.1 encoded certificate that is needed to verify the signature.

Certificate label. The label of the certificate that is needed to verify the signature. This is the label of the certificate in the *OBJECTSIGNING certificate store on the local system. This certificate will need to be exported to the system that will verify this signature.

Distinguished name. The distinguished name of the certificate that is needed to verify the signature.

Length of certificate. Number of bytes needed to contain the ASN.1 encoded certificate.

Length of certificate label. Number of bytes needed to contain the certificate label.

Length of distinguished name. Number of bytes needed to contain the distinguished name.

Length of signature. Number of bytes needed to contain the signature.

Offset to start of certificate. Offset from the beginning of this structure to the certificate.

Offset to start of certificate label. Offset from the beginning of this structure to the certificate label.

Offset to start of distinguished name. Offset from the beginning of this structure to the distinguished name.

Offset to start of signature. Offset from the beginning of this structure to the signature.

Signature. The encrypted hash of the bytestream that was passed in to this API. This can be used later to see if the bytestream has changed.

Error Messages

Message ID	Error Message Text
CPFB724 E	Option &2 of the operating system is required to work with object signatures.
CPFB731 E	Certificate store not found.
CPFB735 E	The digital signing API parameter &1 is not large enough.
CPFB736 E	The digital signing API parameter &1 is not small enough.
CPFB737 E	The digital signing API parameter &1 is a null pointer.
CPFB738 E	The digital signing API parameter &1 is not a valid format type.
CPFB739 E	The digital signing API parameter &1 is out of range.
CPFB73A E	The password for the certificate key database needs to be set.
CPFB73F E	The signing application certificate is expired.
CPFB74A E	The application identifier on the digital signing API is not in a valid state.
CPF9EA0 E	Length of resulting signature area is too small to hold results.
CPF9EAF E	Attempt to sign or verify buffers failed with unexpected return code &1.

API introduced: V5R2

Sign Object (QYDOSGNO, QydoSignObject) API

Required Parameter Group:

1	Object path name	Input	Char(*)
2	Length of object path name	Input	Binary(4)

3	Format of object path name	Input	Char(8)
4	Application identifier	Input	Char(*)
5	Length of application identifier	Input	Binary(4)
6	Replace duplicate signature	Input	Char(1)
7	Multiple objects characteristics	Input	Char(*)
8	Length of multiple objects characteristics	Input	Binary(4)
9	Error code	I/O	Char(*)

Service Program Name: QYDOSGN1
Default Public Authority: *USE
Threadsafe: No

The Sign Object (OPM, QYDOSGNO; ILE, QydoSignObject) API allows the local system to certify that the object being signed is trustworthy as of the time the object is being signed.

The application identifier will be used to find the certificate needed to sign this object. The certificate will be used later to verify the contents of this object have not changed and this certificate will be reported as having signed this object.

Authorities and Locks

Authority Required

For objects in a library:

- *OBJOPR and *OBJMGT authority to the object
- *OBJOPR and *EXECUTE authority to the library.

For objects in a directory:

- *R and *OBJMGT authority to the object
- *X authority to each directory in the path
  *R for the directory with wildcards (that is, a pattern is specified)
  *RX authority to each subdirectory searched if the subdirectories parameter specifies 1.

To use this API, you must be authorized to the object signing applications function associated with your application identifier through iSeries Navigator's application administration support. The Change Function Usage Information (QSYCHFUI) API, with a function ID of the same name as the application identifier, also can be used to change the list of users that are allowed to use this application identifier.

See the open() API for the authority needed to the results path name. The file is open for append and is created if it does not already exist.

Locks   Object will be locked exclusive no read.

Required Parameter Group

Object path name
INPUT; CHAR(*)

The path name of the object you want to sign. If the object is not in a library, the name may be relative to the current directory or may specify the entire path name. If the object is in a library the name must be in the form '/QSYS.LIB/libname.LIB/objname.objtype' if you are using format OBJN0100 object path naming. For example to sign a program named NEWEMPL in library PAYROLL, the qualified object name would be '/QSYS.LIB/PAYROLL.LIB/NEWEMPL.PGM' if you are using format OBJN0100 object path naming. Also if you are using format OBJN0100 object path naming, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

The path name may contain wildcard characters. '*' represents any number of unknown characters. '?' represents any single unknown character. For example, to specify all the program objects in library MYLIB, using format OBJN0100, you could specify '/QSYS.LIB/MYLIB.LIB/*.PGM'. If you want to sign all signable objects in a library or directory, specify the last part of the path name as simply '*'. For example to sign all signable objects in MYLIB, assuming you are using format OBJN0100, you could specify '/QSYS.LIB/MYLIB.LIB/*'.

If the object is in the QSYS file system, it must an object type *PGM, *SRVPGM, *MODULE, *SQLPKG, *FILE (save file), or *CMD.

Length of object path name
INPUT; BINARY(4)

The length of the object path name. If the format of object path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of object path name is OBJN0100, only the path name itself is included.

Format of object path name
INPUT; CHAR(8)

The format of the object path name parameter.

OBJN0100	The object path name is a simple path name.
OBJN0200	The object path name is an LG-type path name.

Application identifier
INPUT; CHAR(*)

The user-supplied application ID to sign objects with. The application type must be 4 (object signing) and it must be assigned to a valid certificate label. User-supplied application IDs should not preface their application ID with QIBM. User-supplied application IDs should start with the company name to eliminate most problems that involve unique names. Application IDs should use an underscore (_) to separate parts of the name (for example, QIBM_OS400_HOSTSERVER). Also, IDs for related applications should start with the same name (for example, QIBM_DIRSRV_SERVER and QIBM_DIRSRV_REPLICATION).

The following characters are allowed in an application ID. The first character of the application ID must be one of the following:

A-Z	Uppercase A-Z

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

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

Length of application identifier
INPUT; BINARY(4)

The length of the specified application identifier. This length must be a value from 1 to 30.

Replace duplicate signature
INPUT; CHAR(1)

Whether the old signature is left or replaced if a signature using the same certificate as the application identifier above uses is detected.

0	Leave the old signature and report an error.
1	Replace the old signature.

If the object contents have changed since the first time this certificate signed the object, the signature is replaced automatically. This parameter only affects signatures where the content has not changed.

Multiple objects characteristics
INPUT; CHAR(*)

How multiple objects specified on the object path name parameter are handled. See "Multiple objects characteristics format" for details on the format of this parameter. This field may be NULL if the length of multiple objects characteristics is 0.

Length of multiple objects characteristics
INPUT; BINARY(4)

The length of the specified multiple objects characteristics. This length may be 0 if you want to use the default values for all these characteristics or 1 or greater to indicate how many bytes of the characteristics should be used.

Error code
I/O; CHAR(*)

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

Multiple objects characteristics format

The format of the multiple objects characteristics is shown in the following table. For detailed descriptions of the fields in the tables, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	CHAR(1)	Subdirectories
1	1	CHAR(1)	Stop of first error
2	2	CHAR(1)	Sign only core part of object
3	3	CHAR(5)	Reserved
8	8	BINARY(4)	Offset to results file path name
12	0C	BINARY(4)	Length of results file path name
16	10	CHAR(8)	Format of results file path name
24	18	CHAR(8)	Format of contents of the results file
		CHAR(*)	Results file path name

Field Descriptions

Format of content of the results file. The format of the content of the file containing the results of this call.

RSLT0100   The basic information is returned for each object specified by the object path name parameter.

Format of results path name. The format of the results path name parameter.

OBJN0100   The results path name is a simple path name.

OBJN0200   The results path name is an LG-type path name.

Length of results path name. The length of the results path name. 0 length means no results files are used, and the results path name and format of results path name parameter values are not used. If the format of results path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of results path name is OBJN0100, only the path name itself is included.

Offset to results path name. Offset from the beginning of this structure to the results path name.

Reserved. This field currently is not used. It is filled with binary zeroes.

Results path name. The path name of the object you want to contain the results on this call. This object may not be in a library (that is, it may not be under the /QSYS.LIB directory). The name may be relative to the current directory or may specify the entire path name. For example, to store results in a file called SIGNED.LST in the MYDIR directory, the results path name would be '/MYDIR/SIGNED.LST'. If you are using format OBJN0100, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

If this is an existing file, results are appended to the end of the file. Otherwise, a new file is created.

The default is not to have a results file.

Sign only core part of object. Whether the entire object be signed or not. This value only applies to objects that can have the core part of the object signed. Objects which cannot have only a core part of the object signed will sign the entire object, independent of the value specified here.

Currently, only *CMD objects can have a core part of the object signed.

0   The entire object should be signed. This is the default value.
1   Only the core part of the object should be signed.

A value of hex 00 will be treated as the default value for this field. This can happen when a program written in V5R1 (where this field was not defined) is run on V5R2.

Stop on first error. Whether control should be returned on the first error found.

0   Continue processing objects even if some errors are found.
1   Stop on the first object that detects an error. This is the default value.

Subdirectories. Whether objects in directories under the directory specified in the object path name parameter should be processed also.

0   Process objects in the directory specified in the object path name parameter only. This is the default value.
1   Process objects in the directory specified in the object name path parameter and in all directories under that directory.

RSLT0100 format

The following table describes the order and format of the data returned in the RSLT0100 format. This data is repeated for each object that was attempted to be processed. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 171.

Note: All data in this file will be in CCSID 13488. New files will be created in this CCSID. If an existing file is named that has a different CCSID, an error will be reported.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(7)	Message identifier
7	7	CHAR(9)	Reserved
16	10	CHAR(8)	Date
24	18	CHAR(8)	Reserved
32	20	CHAR(1)	Operation type
33	21	CHAR(15)	Operation type description
48	30	CHAR(8)	Reserved
56	38	CHAR(*)	Fully qualified object name

Field Descriptions

Date. The date the operation took place. The format will be YYYYMMDD. For example, June 30, 2002 will be 20020630.

Fully qualified object name. The simple path name from the root to the object being signed. The field will be terminated with a new line character.

Message identifier. The error message used to report failure. This field is blank if no error was detected for this object.

Operation type. The operation that was attempted.

0   Signing operation
1   Verifying operation

Operation type description. Short word description of the operation that was attempted.

Reserved. This field currently is not used. It is filled with blanks.

Error Messages

Message ID	Error Message Text
CPF9803 E	Cannot allocate object &2 in library &3.
CPFA085 E	Home directory not found for user &1.
CPFA086 E	Matching quote not found in path name.
CPFA087 E	Path name contains null character.
CPFA088 E	Path name pattern not valid.
CPFA089 E	Pattern not allowed in path name.
CPFA08B E	Path name cannot begin with *.
CPFA08C E	Pattern not allowed in path name directory.
CPFA08D E	Request information value is not valid.
CPFA08E E	More than one name matches pattern.
CPFA091 E	Pattern not allowed in user name.
CPFA092 E	Path name not converted.
CPFA094 E	Path name not specified.
CPFA09C E	Not authorized to object.

Message ID	Error Message Text
CPFA0A4 E	Too many open files for process.
CPFA0AA E	Error occurred while attempting to obtain space.
CPFA0D4 E	File system error occurred.
CPFB720 E	No signable object was found.
CPFB721 E	Object supports signing, but *TGTRLS prevents signing.
CPFB724 E	Option &2 of the operating system is required to work with object signatures.
CPFB72B E	Object not found.
CPFB72C E	The object cannot currently be signed or verified.
CPFB72E E	The parameter for replace duplicate signature is not valid.
CPFB731 E	Sign object certificate database does not exist.
CPFB735 E	The digital signing API parameter &1 is not large enough.
CPFB736 E	The digital signing API parameter &1 is not small enough.
CPFB737 E	The digital signing API parameter &1 is a null pointer.
CPFB738 E	The digital signing API parameter &1 is not a valid format type.
CPFB739 E	The digital signing API parameter &1 is out of range.
CPFB73A E	The password for the certificate key database needs to be set.
CPFB73F E	The signing application certificate is expired.
CPFB740 E	The format name for the pathname is not valid.
CPFB741 E	The length of the path name parameter is not valid.
CPFB742 E	The subdirectory option is an invalid value.
CPFB743 E	The value for stopping on the first error is not valid.
CPFB744 E	The format of the results file for the digital signing API is an incorrect value.
CPFB745 E	The format name for the results file path name is not valid.
CPFB746 E	The results file path name length is not large enough.
CPFB747 E	Object is in a state which is not eligible to be signed.
CPFB748 E	Object signed by IBM, not eligible to be signed.
CPFB749 E	Object signature operation ended abnormally. &3 objects attempted, &2 objects successfully processed.
CPFB74A E	The application identifier on the digital signing API is not in a valid state.
CPFB74C E	Object contains no data to sign (it is empty).
CPFB74D E	Results file could not be used.
CPFBC50 E	No path names match input path names.

API introduced: V5R1

Update User Application Information (QsyUpdateUserApplicationInfo) API

≫

Syntax for QsyUpdateUserApplicationInfo:

```
#include <qsyusrin.h>

void QsyUpdateUserApplicationInfo
      (char           *User_profile,
       char           *Application_information_ID,
       int            *Length_of_application_information_ID,
       char           *Application_information,
       int            *Length_of_application_information,
       char           *First_valid_release,
       void           *Error_code);
```

Service Program: QSYUSRIN
Default Public Authority: *USE
Threadsafe: No

«

The Update User Application Information (QsyUpdateUserApplicationInfo) API updates the specified application information for a user profile. The specified information is stored in an object that is saved and restored with the user profile.

The Change User Profile exit programs are not called from this API.

Authorities and Locks

If the user profile parameter is not *CURRENT or the user profile currently running, then the user profile that calls this API must have *SECADM special authority and *OBJMGT and *USE authorities to the user profile.

Required Parameter Group

User profile
>   INPUT; CHAR(10)

>   The user profile for which the application information will be updated. The special value *CURRENT may be specified to update application information for the user profile that calls this API.

Application information ID
>   INPUT; CHAR(*)

>   The ID for the application information entry to update. IBM-supplied i5/OS application information IDs are named QIBM_*ccc_name*, where *ccc* is the component identifier. User-supplied application information IDs should not preface their application information ID with QIBM. User-supplied application information IDs should start with the company name to eliminate most problems that involve unique names. Application information IDs should use an underscore (_) to separate parts of the name. Also, IDs for related applications should start with the same name.

>   The first character of the application information ID must be one of the following:

A-Z   Uppercase A-Z

>   The remaining characters in the application information ID must be made up of the following characters:

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

Length of application information ID
>   INPUT; BINARY(4)

>   The length of the application information ID that is specified in the application information ID parameter. The length of the application information ID must be a value from 1 to 200.

Application information
>   INPUT; CHAR(*)

>   The application information to be associated with the specified user profile.

Length of application information
> INPUT; BINARY(4)

> The length of the application information that is specified in the application information parameter. The length of the application information must be a value from 1 to 1700.

First valid release
> INPUT; CHAR(6)

> The first release that this application information is valid. This field is used to determine the earliest release this user application information is valid when saving a user profile to a previous release. If the user profile is saved to a release previous to the release specified in this field, this information will not be saved with the user profile information. This field must be in the format VxRxMx (for example, V5R3M0). The release specified must be V5R3M0 or greater.

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.
CPF2213 E	Not able to allocate user profile &1.
CPF2222 E	Storage limit is greater than specified for user profile &1.
CPF226C E	Not authorized to perform function.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF4AA0 E	Application information ID &1 not valid.
CPF4AA1 E	First release value &1 not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R3

Verify Buffer (QYDOVFYB, QydoVerifyBuffer)

Required Parameter Group:

1	Buffer to verify	Input	Char(*)
2	Description of buffer to verify	Input	Char(*)
3	Number of descriptions to verify	Input	Binary(4)
4	Signature to verify	Input	Char(*)
5	Length of signature to verify	Input	Binary(4)
6	Certificate to verify signature	Input	Char(*)
7	Length of certificate to verify signature	Input	Binary(4)
8	Format of the certificate	Input	Char(8)
9	Error Code	I/O	Char(*)

Service Program Name: QYDOBUFFER
Default Public Authority: *USE
Threadsafe: No

The Verify Buffer (OPM, QYDOVFYB; ILE, QydoVerifyBuffer) API allows the local system to verify that the series of bytes signed earlier has not been tampered with. It does this by verifying a digital signature for those bytes.

Authorities and Locks

API Public Authority
> *USE.

Authority Required
> None.

Required Parameter Group

Buffer to verify
> INPUT; CHAR(*)

> The buffer of data to be verified. Only the part of the object described in the Description of buffer to sign will be verified.

Description of buffer to verify
> INPUT; CHAR(*)

> Array of offsets and lengths to the data to be verified. The API will treat these bytes as if they were a contiguous stream of bytes.

> The format of the description of the data to verify is in the following table. For detailed descriptions of the fields in this table, see "Field Descriptions" on page 176.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	Binary(4)	Offset to start of first series of bytes to verify
4	4	Binary(4)	Length of first series of bytes to verify
n	n	Binary(4)	Offset to start of next series of bytes to verify
n+4	n+4	Binary(4)	Length of next series of bytes to verify

Number of descriptions to verify
> INPUT; BINARY(4)

> Number of offsets and lengths needed to describe data to be verified.

Signature to verify
> INPUT; CHAR(*)

> The signature to be verified. This signature will be checked against the data identified in the first two parameters to see if the data has changed since it was signed by this signature.

Length of signature to verify
> INPUT; BINARY(4)

> Length of the specified signature.

Certificate to verify signature
    INPUT; CHAR(*)

    The certificate that was used to create the signature. This certificate must be in the format described in the 'Format of the certificate' parameter.

Length of certificate to verify signature
    INPUT; BINARY(4)

    Length of the specified certificate.

Format of the certificate
    INPUT; CHAR(8)

    The format of the certificate to verify parameter:

CERT0100	Certificate label. Uses *SIGNATUREVERIFICATION certificate store to find certificate.
CERT0200	ASN.1 encoded certificate. This is the certificate itself.
CERT0300	Distinguished name of certificate. Uses LDAP server to find certificate.

Error code
    I/O; CHAR(*)

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

Field Descriptions

Length of first series of bytes to verify The number of bytes, including the first byte in the series, to be included in the signature.

Length of next series of bytes to verify The number of bytes, including the first byte in the series, to be included in the signature.

Offset to start of first series of bytes to verify. An offset to the first byte of a series of 1 or more bytes of data to be included in the signature.

Pointer to start of next series of bytes to verify. An offset to the first byte of a series of 1 or more bytes of data to be included in the signature.

Error Messages

Message ID	Error Message Text
CPFB724 E	Option &2 of the operating system is required to work with object signatures.
CPFB731 E	Certificate store not found.
CPFB735 E	The digital signing API parameter &1 is not large enough.
CPFB736 E	The digital signing API parameter &1 is not small enough.
CPFB737 E	The digital signing API parameter &1 is a null pointer.
CPFB738 E	The digital signing API parameter &1 is not a valid format type.
CPFB739 E	The digital signing API parameter &1 is out of range.
CPFB73A E	The password for the certificate key database needs to be set.
CPF9EA0 E	Length of resulting signature area is too small to hold results.
CPF9EA1 E	Signature parameter is not in a supported format.
CPF9EA0 E	Length of resulting signature area is too small to hold results.

Message ID	Error Message Text
CPF9EA2 E	Certificate is not in a supported format.
CPF9EA3 E	Certificate with label &2 not found.
CPF9EA4 E	Buffer has a signature that is not valid..
CPF9EAF E	Attempt to sign or verify buffers failed with unexpected return code &1.

Introduced: V5R2

Verify Object (QYDOVFYO, QydoVerifyObject) API

Required Parameter Group:

1	Object path name	Input	Char(*)
2	Length of object path name	Input	Binary(4)
3	Format of object path name	Input	Char(8)
4	Multiple objects characteristics	Input	Char(*)
5	Length of multiple objects characteristics	Input	Binary(4)
6	Error code	I/O	Char(*)

Service Program Name: QYDOVFY1
Default Public Authority: *USE
Threadsafe: No

The Verify Object (OPM, QYDOVFYO; ILE, QydoVerifyObject) API checks to see if an object has changed since it was signed. Only certificates in the local system's Verify Object certificate database that have signed this object will be checked. Any other signatures will be ignored. If none of the signatures of this object are by certificates the local system recognizes, the object is considered unsigned. If the object is unsigned, this is reported as an error. If any trusted signatures are valid, the object is considered successfully verified.

Authorities and Locks

Authority Required
> *AUDIT special authority is optional; if used, all objects can be verified. If *AUDIT special authority is not used, you need to have:

> For objects in a library:
> - *READ authority to the object
> - *OBJOPR and *EXECUTE authority to the library.

> For objects in a directory:
> - *R authority to the object
> - *X authority to each directory in the path
>   *R for the directory with wildcards (that is, a pattern is specified)
>   *RX authority to each subdirectory searched if the subdirectories parameter specifies 1.

> See the open() API for the authority needed to the results path name. The file is open for append and is created if it does not already exist.

Locks Object will be locked shared allow read. Certificate database will be locked while certificates are retrieved (to make up trusted certificate list needed to verify).

Required Parameter Group

Object path name
      INPUT; CHAR(*)

The name of the object you want to verify. If the object is not in a library, the name may be relative to the current directory or may specify the entire path name. If the object is in a library the name must be in the form '/QSYS.LIB/libname.LIB/objname.objtype' if you are using format OBJN0100 object path naming. For example to sign a program named NEWEMPL in library PAYROLL, the qualified object name would be '/QSYS.LIB/PAYROLL.LIB/NEWEMPL.PGM' if you are using format OBJN0100 object path naming. Also if you are using format OBJN0100 object path naming, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

The path name may contain wildcard characters. '*' will represent any number of unknown characters. '?' will represent any single unknown character. For example, to specify all the program objects in library MYLIB, using format OBJN0100, you could specify '/QSYS.LIB/MYLIB.LIB/*.PGM'. If you want to verify all signable objects in a library or directory, specify the last part of the path name as simply '*'. For example to verify all signable objects in MYLIB, assuming you are using format OBJN0100, you could specify '/QSYS.LIB/MYLIB.LIB/*'.

If the object is in the QSYS file system, it must an object type *PGM, *SRVPGM, *MODULE, *SQLPKG, *FILE (save file), or *CMD.

Length of object path name
      INPUT; BINARY(4)

The length of the object path name. If the format of object path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of object path name is OBJN0100, only the path name itself is included.

Format of object path name
      INPUT; CHAR(8)

The format of the object path name parameter.

OBJN0100	The object path name is a simple path name.
OBJN0200	The object path name is an LG-type path name.

Multiple objects characteristics
      INPUT; CHAR(*)

How multiple objects specified on the object path name parameter are handled. See "Multiple objects characteristics format" on page 179 for details on the format of this parameter. This field may be NULL if the length of multiple objects characteristics is 0.

Length of multiple objects characteristics
      INPUT; BINARY(4)

The length of the specified multiple objects characteristics. This length may be 0 if you want to use the default values for all these characteristics, or 1 or greater to indicate how many bytes of the characteristics should be used.

Error code
      I/O; CHAR(*)

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

Multiple objects characteristics format

The format of the multiple objects characteristics is shown in the following table. For detailed descriptions of the fields in the tables, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	CHAR(1)	Subdirectories
1	1	CHAR(1)	Stop of first error
2	2	CHAR(6)	Reserved
8	8	BINARY(4)	Offset to results file path name
12	0C	BINARY(4)	Length of results file path name
16	10	CHAR(8)	Format of results file path name
24	18	CHAR(8)	Format of contents of the results file
		CHAR(*)	Results file path name

Field Descriptions

Format of content of the results file. The format of the contents of the file containing the results of this call.

RSLT0100   The basic information is returned for each object specified by the object path name parameter.

Format of results path name. The format of the results path name parameter.

OBJN0100   The results path name is a simple path name.
OBJN0200   The results path name is an LG-type path name.

Length of results path name. The length of the results path name. 0 length means no results files are used, and the results path name and format of results path name parameter values are not used. If the format of results path name is OBJN0200, this field must include the QLG path name structure in addition to the path name itself. If the format of results path name is OBJN0100, only the path name itself is included.

Offset to results path name. Offset from the beginning of this structure to the results path name.

Reserved. This field currently is not used. It is filled with binary zeroes.

Results path name. The path name of the object you want to contain the results on this call. This object may not be in a library (that is, may not be under the /QSYS.LIB directory). The name may be relative to the current directory or may specify the entire path name. For example to store results in a file called SIGNED.LST in the MYDIR directory, the results path name would be '/MYDIR/SIGNED.LST'. If you are using format OBJN0100, this parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job.

If this is an existing file, results will be appended to the end of the file. Otherwise, a new file will be created.

The default is not to have a results file.

Stop on first error. Whether control should be returned on the first error found.

0   Continue processing objects even if some errors are found.
1   ″Start of change″>Stop on the first object that detects an error. This is the default value.

Subdirectories. Whether objects in directories under the directory specified in the object path name parameter should be processed also.

0   Process objects in the directory specified in the object path name parameter only. This is the default value.
1   Process objects in the directory specified in the object name path parameter and in all directories under that directory.

RSLT0100 format

The following table describes the order and format of the data returned in the RSLT0100 format. This data is repeated for each object that was attempted to be verified. For detailed descriptions of the fields in the tables, see "Field Descriptions."

Note: All data in this file will be in CCSID 13488. New files will be created in this CCSID. If an existing file is named that has a different CCSID, an error will be reported.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(7)	Message identifier
7	7	CHAR(9)	Reserved
16	10	CHAR(8)	Date
24	18	CHAR(8)	Reserved
32	20	CHAR(1)	Operation type
33	21	CHAR(15)	Operation type description
48	30	CHAR(8)	Reserved
56	38	CHAR(*)	Fully qualified object name

Field Descriptions

Date. The date the operation took place. The format will be YYYYMMDD. For example, June 30, 2002 will be 20020630.

Fully qualified object name. The simple path name from the root to the object whose signature is being verified. The field will be terminated with a new line character.

Message identifier. The error message used to report failure. This field is blank if no error was detected for this object.

Operation type. The operation that was attempted.

0   Signing operation
1   Verifying operation

Operation type description. Short word description of the operation that was attempted.

Reserved. This field currently is not used. It is filled with blanks.

Error Messages

Message ID	Error Message Text
CPFA085 E	Home directory not found for user &1.
CPFA086 E	Matching quote not found in path name.
CPFA087 E	Path name contains null character.
CPFA088 E	Path name pattern not valid.
CPFA089 E	Pattern not allowed in path name.
CPFA08B E	Path name cannot begin with *.
CPFA08C E	Pattern not allowed in path name directory.
CPFA08D E	Request information value is not valid.
CPFA08E E	More than one name matches pattern.
CPFA091 E	Pattern not allowed in user name.
CPFA092 E	Path name not converted.
CPFA094 E	Path name not specified.
CPFA0A4 E	Too many open files for process.
CPFA0AA E	Error occurred while attempting to obtain space.
CPFA0D4 E	File system error occurred.
CPFB720 E	No signable object was found.
CPFB722 E	Object not signed.
CPFB723 E	Object signed, but signature is not valid.
CPFB724 E	Option &2 of the operating system is required to work with object signatures.
CPFB72A E	The object had no trusted signatures on the object.
CPFB72B E	Object not found.
CPFB72C E	The object cannot currently be signed or verified.
CPFB735 E	The digital signing API parameter &1 is not large enough.
CPFB736 E	The digital signing API parameter &1 is not small enough.
CPFB737 E	The digital signing API parameter &1 is a null pointer.
CPFB738 E	The digital signing API parameter &1 is not a valid format type.
CPFB739 E	The digital signing API parameter &1 is out of range.
CPFB73A E	The password for the certificate key database needs to be set.
CPFB740 E	The format name for the pathname is not valid.
CPFB741 E	The length of the path name parameter is not valid.
CPFB742 E	The subdirectory option is an invalid value.
CPFB743 E	The value for stopping on the first error is not valid.
CPFB744 E	The format of the results file for the digital signing API is an incorrect v.
CPFB745 E	The format name for the results file path name is not valid.
CPFB746 E	The results file path name length is not large enough.
CPFB749 E	Object signature operation ended abnormally. &1 objects attempted, &2 objects successfully processed.
CPFB74D E	Results file could not be used.
CPFBC50 E	No path names match input path names.

API introduced: V5R1

Security-related Exit Programs

You may write exit programs that are called by the operating system to perform user-profile-related functions that suit your needs. With the use of these exit programs, the user-profile exit points notify you when a user profile has been created, changed, and so forth. For example, if you are maintaining a network of systems and you want to keep the user profile changes synchronized on all the systems, you can use these exit points to be notified of all the create, change, and delete profile activity. When a user profile is created on one system and you receive notification of that user profile being created, you can

retrieve all the user profile information and create a duplicate user profile on the other systems in your network. The same can be done for any user profile changes and deletions.

For general information abou the i5/OS^(TM) system security, see the iSeries Security Reference ![book icon] book.

The i5/OS security-related exit programs are:
- "Change User Profile Exit Program" is called when a user profile has been changed on the iSeries.
- » "Check Password Exit Program" on page 184 (QIBM_QSY_CHK_PASSWRD) is called when a command or the QSYCHGPW API is used to set or change the password associated with a user profile. «
- "Create User Profile Exit Program" on page 187 is called when a user profile is created on the iSeries.
- "Delete User Profile Exit Program" on page 188 is called when a user profile is deleted on the iSeries.
- "Restore User Profile Exit Program" on page 189 is called when a user profile is restored on the iSeries.
- "Validate Password Exit Program" on page 190 is called when a Change Password (CHGPWD) command or Change Password (QSYCHGPW) API is executed.

Note: The QIBM_QSY_HOSTFUNC, QIBM_QSY_OPNAVCENTRL, QIBM_QSY_OPNAVCLIENT, QIBM_QSY_OTHERCENTRL, and QIBM_QSY_OTHERCLIENT exit points are used only to store function registration information (see the Register Function (QSYRGFN, QsyRegisterFunction) API). 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.

Exit Programs

These are the Exit Programs for this category.

Change User Profile Exit Program

Required Parameter:

1	Change profile exit information	Input	Char(*)

»
```
QSYSINC Member Name:  ECHGPRF1
Exit Point Name:  QIBM_QSY_CHG_PROFILE
Exit Point Format Name:  CHGP0100, CHGP0200
```

«

» The Change User Profile exit programs are called when changing a user profile using one of the following commands or API: «
- Change User Profile (CHGUSRPRF) command
- Change User Auditing (CHGUSRAUD) command
- Reset Profile Attributes (QSYRESPA) API

Other i5/OS commands and APIs call the interfaces listed above. As a result, they cause the Change User Profile exit programs to be called. A partial list of additional i5/OS interfaces that force calls to the exit programs are listed below:
- Change Profile (CHGPRF) command

- Change Password (CHGPWD) command
- Change Password (QSYCHGPW) API
- Configure System Security (CFGSYSSEC) command
- Analyze Profile Activity (ANZPRFACT) command

There are some i5/OS interfaces that make changes to the user profile object that do not cause the Change User Profile exit programs to be called. The most notable interfaces are listed below:

- Changing the user profile text description with the Change Object Description (CHGOBJD) command
- Granting or revoking private authority to an object
- Changing object ownership information
- Disabling a user profile during sign-on
- Setting the encrypted user password with the Set Encrypted Password (QSYSUPWD) API

The functions which do or don't cause the exit program to be called may change from release to release as commands and APIs are added.

≫ When changing a user profile, as explained above, the operating system calls the user-written exit programs through the registration facility. Exit programs can register to be notified before the profile is changed, after the profile is changed, or both. The prechange notification is sent prior to actually changing the profile. The prechange notification is not a guarantee that the profile will actually be changed. The postchange notification is sent after the profile is changed.

It is also possible for a prechange notification to be sent for more than one profile before receiving a postchange notification. Therefore, an application that wants to keep a prechange view of the profile to compare to the postchange view of the profile needs to be able to handle prechange views for multiple profiles and be able to match those views with the correct profile during postchange processing.

For example, this could occur during CHGUSRPRF processing if the group profile or supplemental group list is changed. If a profile that is specified as a group profile or supplemental group does not already have a group ID (GID) assigned to the profile, then the CHGUSRPRF code will assign a GID to the profile. This involves a change to that profile, so a prechange notification will be sent for the profile having a GID assigned after the prechange notification for the profile specified on CHGUSRPRF, but before the postchange notification for the profile specified on CHGUSRPRF. ≪

The exit point supports an unlimited number of exit programs. (For information about adding an exit program to an exit point, see the Registration Facility part.)

Note: The Change User Profile exit point ignores any return codes or error messages that are sent from the exit program.

Authorities and Locks

User Profile Authority
> *ALLOBJ and *SECADM to add exit programs to the registration facility

Required Parameter

Change profile exit information
> INPUT; CHAR(*)

> Information needed by the exit program for notification of any profile changes. For details, see "Format of Change Profile Exit Information" on page 184.

Format of Change Profile Exit Information

The following table shows the structure of the change profile exit information for » formats CHGP0100 and CHGP0200. « For a description of the fields in the format, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	CHAR(20)	Exit point name
20	14	CHAR(8)	Exit point format name
28	1C	CHAR(10)	User profile name

Field Descriptions

Exit point format name. The format name for the Change User Profile exit program. » The possible format names are: «

CHGP0100	The format name that is used after a user profile is changed.
»CHGP0200	The format name that is used before a user profile is changed. «

Exit point name. The name of the exit point that calls the exit program.

User profile name. The name of the user profile being changed.

Exit program introduced: V3R7

Check Password Exit Program

Required Parameter:

1	Check password exit information	Input	Char(*)
2	Return indicator	Output	Char(1)

QSYSINC Member Name: ECHKPWD1
Exit Point Name: QIBM_QSY_CHK_PASSWRD
Exit Point Format Name: CHKP0100

The Check Password exit program is called when a Create User Profile (CRTUSRPRF) command, Change User Profile (CHGUSRPRF) command, Change Password (CHGPWD) command, or Change Password (QSYCHGPW) API is used to set or change the password associated with a user profile. The exit program is called after the system value based password composition rules have been successfully checked.

Note: The Limit Password Character Positions (QPWDPOSDIF) system value based composition rule is not checked before calling the exit program because the old password value is not available through all of the interfaces that can set or change the password.

The exit program can examine the new password value for conformance with customer unique password composition rules. The exit program returns an indication whether the new password conforms to the customer's password rules. This indication will be used so that the security audit journal can record whether the changed password conforms to the password composition rules. However, the password will be changed regardless of whether the exit program returns an indication that the password does not conform to the customer's password rules.

The exit point supports multiple exit programs. However, additional exit programs will not be called after receiving an indication that the new password does not conform from one of the exit programs. (For information about adding an exit program to an exit point, see the Registration Facility part.)

Any escape message received from an exit program or encountered while trying to call an exit program, will be treated as an indication that the new password does not conform to the customer's password rules.

The specified exit program must exist in the system auxiliary storage pool (ASP) or one of the basic user ASPs at the time it is added to the registration facility. If the program does not exist, the request to add the exit program will be rejected.

The exit program must exist in the system ASP or one of the basic user ASPs at the time the exit point attempts to locate the exit program. If the specified exit program does not exist in the system ASP or one of the basic user ASPs, the condition will be treated as an indication that the new password does not conform to the customer's password rules.

This exit point is very similar to the QIBM_QSY_VLD_PASSWRD exit point. The key differences between these two exit points are:
- This exit point is called after the password has been changed while the QIBM_QSY_VLD_PASSWRD exit is called before the password is changed.
- The returned indicator for this exit program does not effect the password change while the return indicator from the QIBM_QSY_VLD_PASSWRD exit does effect whether the password is changed.
- This exit point is called for passwords set by the CRTUSRPRF and CHGUSRPRF comands while the QIBM_QSY_VLD_PASSWRD is not called for either of these commands.
- All cases where the QIBM_QSY_VLD_PASSWRD exit is called (and all successful indications were returned) will also call this exit. However, the reverse is not true.

Authorities and Locks

User Profile Authority
> *ALLOBJ and *SECADM to add or remove exit programs to the registration facility

Required Parameter

Check password exit information
> INPUT; CHAR(*)
>
> Information needed by the exit program for notification of a password change. For details, see "Format of Check Password Exit Information."

Return indicator
> OUTPUT; CHAR(1)
>
> Indicates whether the new password conforms to the customer's password rules.

'0'   Indicates that the new password conforms to the customer's password rules.
'1'   Indicates that the new password does not conform to the customer's password rules.

> **Note:** Any value other than '0' indicates that the new password does not conform to the customer's password rules.

Format of Check Password Exit Information

The following table shows the structure of the check password exit information for format CHKP0100. For a description of the fields in this format, see "Field Descriptions" on page 186.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(20)	Exit point name
20	14	CHAR(8)	Exit point format name
28	1C	BINARY(4)	Password level
32	20	CHAR(10)	User profile name
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Offset to old password
48	30	BINARY(4)	Length of old password
52	34	BINARY(4)	CCSID of old password
56	38	BINARY(4)	Offset to new password
60	3C	BINARY(4)	Length of new password
64	40	BINARY(4)	CCSID of new password
		CHAR(*)	Old password
		CHAR(*)	New password

Field Descriptions

CCSID of new password. The CCSID of the new password field. The CCSID value will be 13488.

CCSID of old password. The CCSID of the old password field. The CCSID value will be 13488.

Exit point format name. The format name for the Check Password exit program. The possible format name is:

CHKP0100   The format name that is used after a user password is changed by the CRTUSRPRF, CHGUSRPRF, CHGPWD command or QSYCHGPW API.

Exit point name. The name of the exit point that calls the exit program.

Length of new password. The length, in bytes, of the new password field.

Length of old password. The length, in bytes, of the old password field.

The length value will be 12. The old password value is not available in all cases so a value of '*NOPWD' is used. This allows the format of the information passed to the QIBM_QSY_CHK_PASSWRD and QIBM_QSY_VLD_PASSWRD exit programs to be the same.

New password. The new password value.

Offset to new password. The offset from the beginning of the check password exit information to the new password field.

Offset to old password. The offset from the beginning of the check password exit information to the old password field.

Old password. The old password value.

The old password value is not available in all cases so a value of '*NOPWD' is used. This allows the format of the information passed to the QIBM_QSY_CHK_PASSWRD and QIBM_QSY_VLD_PASSWRD exit programs to be the same.

Password level. The password level in affect for the system. See the QPWDLVL system value for a description of the possible values.

User profile name. The name of the user profile whose password is being changed.

«

API introduced: V5R4

Create User Profile Exit Program

Required Parameter:

1	Create profile exit information	Input	Char(*)

```
QSYSINC Member Name:  ECRTPRF1
Exit Point Name:  QIBM_QSY_CRT_PROFILE
Exit Point Format Name:  CRTP0100
```

The Create User Profile exit program is called when a user profile is created on the iSeries server.

When a user profile is created on the iSeries server, the operating system calls the user-written exit programs through the registration facility. The exit point supports an unlimited number of exit programs. (For information about adding an exit program to an exit point, see the Registration Facility part.)

Note: The Create User Profile exit program ignores any return codes or error messages that are sent from the exit program.

Authorities and Locks

User Profile Authority
  *ALLOBJ and *SECADM to add exit programs to the registration facility

Required Parameter

Create profile exit information
  INPUT; CHAR(*)

  Information needed by the exit program for notification of any profile being created. For details, see "Format of Create Profile Exit Information."

Format of Create Profile Exit Information

The following table shows the structure of the create profile exit information for format CRTP0100. For a description of the fields in this format, see "Field Descriptions" on page 188.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(20)	Exit point name
20	14	CHAR(8)	Exit point format name

Offset			
Dec	Hex	Type	Field
28	1C	CHAR(10)	User profile name

Field Descriptions

Exit point format name. The format name for the Create User Profile exit program. The possible format name is:

CRTP0100   The format name that is used after a user profile created.

Exit point name. The name of the exit point that is calling the exit program.

User profile name. The name of the user profile that was created.

Exit program introduced: V3R7

Delete User Profile Exit Program

Required Parameter:

1   Delete profile exit information   Input   Char(*)

```
QSYSINC Member Name:  EDLTPRF1, EDLTPRF2
Exit Point Name:  QIBM_QSY_DLT_PROFILE
Exit Point Format Names:  DLTP0100, DLTP0200
```

The Delete User Profile exit program is called when a user profile is deleted on the iSeries server.

When a user profile is deleted on the iSeries server, the operating system calls the user-written exit programs through the registration facility. Exit programs can register to be notified before the profile is deleted, after the profile is deleted, or both. The predeletion notification is sent prior to doing any owned objects checking, which is required for the deletion of a user profile. Therefore, the predeletion notification is not a guarantee that the profile will actually be deleted. The postdeletion notification is sent after the profile is deleted.

The exit point supports an unlimited number of exit programs. (For information about adding an exit program to an exit point, see the Registration Facility part.)

Note: The Delete User Profile exit program ignores any return codes or error messages that are sent from the exit program.

Authorities and Locks

User Profile Authority
          *ALLOBJ and *SECADM to add exit programs to the registration facility

Required Parameter

Delete profile exit information
          INPUT; CHAR(*)

Information needed by the exit program for notification of any profile deletions. For details, see "Format of Delete Profile Exit Information."

Format of Delete Profile Exit Information

The following table shows the structure of the delete profile exit information for formats DLTP0100 and DLTP0200. For a description of the fields in this format, see "Field Descriptions."

Offset			
Dec	Hex	Type	Field
0	0	CHAR(20)	Exit point name
20	14	CHAR(8)	Exit point format name
28	1C	CHAR(10)	User profile name

Field Descriptions

Exit point format name. The format name for the Delete User Profile exit program. The possible format name is:

DLTP0100   The format name that is used after a user profile is deleted.
DLTP0200   The format name that is used before a user profile is deleted.

Exit point name. The name of the exit point that is calling the exit program.

User profile name. The name of the user profile being deleted.

Exit program introduced: V3R7

Restore User Profile Exit Program

Required Parameter:

1   Restore profile exit information   Input   Char(*)

```
QSYSINC Member Name:  ERSTPRF1
Exit Point Name:  QIBM_QSY_RST_PROFILE
Exit Point Format Name:  RSTP0100
```

The Restore User Profile exit program is called when a user profile is restored on the iSeries server.

When a user profile is restored to the iSeries server, the operating system calls the user-written exit programs through the registration facility.

During a restore operation of the entire system, all required objects are not installed at the time user profiles are being restored. Therefore, this exit point is not active during a restore operation of the entire system.

This exit point supports up to 20 exit programs. (For information about adding an exit program to an exit point, see the Registration Facility part.)

Note: The Restore User Profile exit program ignores any return codes or error messages that are sent from the exit program.

Authorities and Locks

User Profile Authority
> *ALLOBJ and *SECADM to add exit programs to the registration facility

Required Parameter

Restore profile exit information
> INPUT; CHAR(*)

> Information needed by the exit program for notification of any profile restored. For details, see "Format of Restore Profile Exit Information."

Format of Restore Profile Exit Information

The following table shows the structure of the restore profile exit information for format RST0100. For a description of the fields in this format, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(20)	Exit point name
20	14	CHAR(8)	Exit point format name
28	1C	CHAR(10)	User profile name

Field Descriptions

Exit point format name. The format name for the Restore User Profile exit program. The possible format name is:

RSTP0100   The format name that is used after a user profile is restored.

Exit point name. The name of the exit point that calls the exit program.

User profile name. The name of the user profile that was restored.

Exit program introduced: V3R7

Validate Password Exit Program

Required Parameter:

1	Validate password exit information	Input	Char(*)
2	Return indicator	Output	Char(1)

QSYSINC Member Name: EVLDPWD1
Exit Point Name: QIBM_QSY_VLD_PASSWRD
Exit Point Format Name: VLDP0100

The Validate Password exit program is called when a Change Password (CHGPWD) command or Change Password (QSYCHGPW) API is executed. The exit program is called after the password composition rules have been checked.

The exit program examines the old and new password values for conformance with customer unique password composition rules. The exit program returns an indication whether the new password should be accepted or rejected. The exit point supports multiple exit programs. However, additional exit programs will not be called after receiving a indication that the new password should be rejected from one of the exit programs. (For information about adding an exit program to an exit point, see the Registration Facility part.)

Any escape message received from an exit program or encountered while trying to call an exit program, will be treated as an indication that the new password should be rejected.

The specified exit program must exist in the system auxiliary storage pool (ASP) or one of the basic user ASPs at the time it is added to the registration facility. If the program does not exist, the request to add the exit program will be rejected.

The exit program must exist in the system ASP or one of the basic user ASPs at the time the exit point attempts to locate the exit program. If the specified exit program does not exist in the system ASP or one of the basic user ASPs, the condition will be treated as an indication that the new password should be rejected.

Note: The QPWDVLDPGM system value must be set to the value *REGFAC. If the QPWDVLDPGM system value contains any other value, the validate password exit programs will not be called.

Authorities and Locks

User Profile Authority
> *ALLOBJ and *SECADM to add or remove exit programs to the registration facility

Required Parameter

Validate password exit information
> INPUT; CHAR(*)
>
> Information needed by the exit program for notification of any profile changes. For details, see "Format of Validate Password Exit Information."

Return indicator
> OUTPUT; CHAR(1)
>
> Indicates whether the new password should be accepted or rejected.

'0'   Indicates that the new password should be accepted.
'1'   Indicates that the new password should be rejected.

> **Note:** Any value other than '0' indicates that the new password should be rejected.

Format of Validate Password Exit Information

The following table shows the structure of the validate password exit information for format VLDP0100. For a description of the fields in this format, see "Field Descriptions" on page 192.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(20)	Exit point name

Offset		Type	Field
Dec	Hex		
20	14	CHAR(8)	Exit point format name
28	1C	BINARY(4)	Password level
32	20	CHAR(10)	User profile name
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Offset to old password
48	30	BINARY(4)	Length of old password
52	34	BINARY(4)	CCSID of old password
56	38	BINARY(4)	Offset to new password
60	3C	BINARY(4)	Length of new password
64	40	BINARY(4)	CCSID of new password
		CHAR(*)	Old password
		CHAR(*)	New password

Field Descriptions

CCSID of new password. The CCSID of the new password field. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

CCSID of old password. The CCSID of the old password field. For a list of valid CCSIDs, see the Globalization topic in the iSeries Information Center.

Exit point format name. The format name for the Change User Profile exit program. The possible format name is:

VLDP0100　　　　The format name that is used before a user password is changed by the CHGPWD command or QSYCHGPW API.

Exit point name. The name of the exit point that calls the exit program.

Length of new password. The length, in bytes, of the new password field.

When called by the QSYCHGPW API, this is the length supplied to (or defaulted to) the QSYCHGPW API. It may include trailing blank or null characters which are removed by the system before changing the password.

When called by the CHGPWD command, this is the length of the actual password with any trailing blank or null characters removed.

Length of old password. The length, in bytes, of the old password field.

When called by the QSYCHGPW API, this is the length supplied to (or defaulted to) the QSYCHGPW API. It may include trailing blank or null characters which are removed by the system before changing the password.

When called by the CHGPWD command, this is the length of the actual password with any trailing blank or null characters removed.

New password. The new password value.

Offset to new password. The offset from the beginning of the validate password exit information to the new password field.

Offset to old password. The offset from the beginning of the validate password exit information to the old password field.

Old password. The old password value.

Password level. The password level in affect for the system. See the QPWDLVL system value for a description of the possible values.

User profile name. The name of the user profile whose password is being changed.

API introduced: V3R1

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:

195

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.

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.

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:

(C) IBM 2006. Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1998, 2006. All rights reserved.

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

Programming Interface Information

This Application Programming Interfaces (API) 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 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
i5/OS
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

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

IBM ®

Printed in USA