# Miscellaneous APIs (V5R2)

## Table of Contents

# Miscellaneous APIs

The miscellaneous APIs are APIs that do not logically fall in a specific part of the OS/400 reference information.

The miscellaneous APIs consist of:

- General Miscellaneous APIs
- Process Open List APIs

---

APIs by category

# General Miscellaneous APIs

The general miscellaneous APIs perform a variety of functions, including removing bookmarks from a course, converting date and time, starting pass-through, and retrieving data on a target system.

The general miscellaneous APIs are:

- Add Seed for Pseudorandom Number Generator (Qc3AddPRNGSeed) allows the user to add seed into the server's pseudorandom number generator system seed digest.
- Check Communications Trace (QSCCHKCT) returns, in bytes, the maximum size configured for the communications trace tool and the portion of that size that is currently in use for all communications traces active (running or stopped state), or zero if no traces are active.
- Control Device (QTACTLDV) provides a direct command interface to a device.
- Convert Date and Time Format (QWCCVTDT) allows you to convert date and time formats from one format to another format.
- Convert Timeval Structure to _MI_Time (Qp0zCvtToMITime()) converts a UNIX-type timestamp (or timestamp offset), represented by a timeval structure, to a corresponding _MI_Time data type.
- Convert _MI_Time to Timeval Structure (Qp0zCvtToTimeval()) converts a machine timestamp (or timestamp offset), represented by an _MI_Time data type, to a corresponding timeval structure.
- Generate Pseudorandom Numbers (Qc3GenPRNs) generates a pseudorandom binary stream.
- Remove All Bookmarks from a Course (QEARMVBM) allows you to remove the bookmarks from a Tutorial System Support course.
- Retrieve Data (QPARTVDA) retrieves up to 1KB of user data, which was passed to this system with the Start Pass-through (QPASTRPT) API.
- Start Pass-Through (QPASTRPT) starts a 5250 pass-through session and optionally passes up to 1KB of user data from the source system to the target system. This data can be accessed on the target system with the Retrieve Data (QPARTVDA) API.

The exit program within the general miscellaneous APIs is:

- Device Selection provides an interface to control virtual device selection and automatic creation used by the system for connection requests from clients using virtual device support.

---

# Add Seed for Pseudorandom Number Generator (Qc3AddPRNGSeed) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Seed data | Input | Char(*) |
| 2 | Seed data length | Input | Binary(4) |
| 3 | Error Code | I/O | Char(*) |

Service Program Name: QC3PRNG

Default Public Authority: *USE

Threadsafe: Yes

The Add Seed for Pseudorandom Number Generator (Qc3AddPRNGSeed) API allows the user to add seed into the server's pseudorandom number generator system seed digest.

The pseudorandom number generator is composed of two parts: pseudorandom number generation and seed management. Pseudorandom number generation is performed using the FIPS 186-1 algorithm. (See the Generate Pseudorandom Numbers (Qc3GenPRNs) API.) Cryptographically-secure pseudorandom numbers rely on good seed. The FIPS 186-1 key and seed values are obtained from the system seed digest. The server automatically generates seed using data collected from system information or by using the random number generator function on a cryptographic coprocessor, such as a 4758, if one is available. System-generated seed can never be truly unpredictable. If a cryptographic coprocessor is not available, you can use this API to add your own random seed to the system seed digest. This should be done as soon as possible any time the Licensed Internal Code is installed.

## Authorities and Locks

All object (*ALLOBJ) special authority is needed to use this API.

*User Profile Authority*

*ALLOBJ

## Required Parameter Group

**Seed data**

INPUT; CHAR(*)

The input seed data for the system seed digest.

It is important that the seed data be unpredictable and have as much entropy as possible. Entropy is the minimum number of bits needed to represent the information contained in some data. For seeding purposes, entropy is a measure of the amount of uncertainty or unpredictability of the seed. The system seed digest holds a maximum of 160 bits of entropy. You should add at least that much entropy to refresh the system seed digest totally. Possible sources of seed data are coin flipping, keystroke or mouse timings, or a noise source such as the one available on the 4758 Cryptographic Coprocessor.

**Seed data length**

INPUT; BINARY(4)

The length of the seed data, in bytes. If this length is 0, no seed data is added.

**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 | *ALLOBJ special authority is required. |
| CPF3C17 E | Error occurred with input data parameter. |
| CPF3CF1 E | Error code parameter not valid. |

---

API introduced: V5R1

---

# »Check Communications Trace (QSCCHKCT) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Storage allocated | Output | Binary(8) |
| 2 | Storage in use | Output | Binary(8) |
| 3 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Check Comunications Trace (QSCCHKCT) API returns, in bytes, the maximum size configured for the communications trace tool and the portion of that size that is currently in use for all communications traces active (running or stopped state), or zero if no traces are active.

## Authorities and Locks

Caller must have *SERVICE special authority, or be authorized to the Service Trace function of OS/400 through iSeries Navigator's Application Administration support.

## Required Parameter Group

**Storage allocated**

OUTPUT; Binary(8)

The variable containing the maximum bytes configured for the communications trace tool after the QSCCHKCT API has completed processing.

**Storage in use**

Output; Binary(8)

The variable containing the total bytes in use for all communications traces active (running or stopped state), or zero if no traces are active, after the QSCCHKCT API has completed processing.

**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. |
| CPF39A8 E | Not authorized to communications trace service tool. |
| CPF39B6 E | Communications trace function cannot be performed. |
| CPF3CF2 E | Error(s) occurred during running of &1 API. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V5R2

# Control Device (QTACTLDV) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Device name | Input | Char(10) |
| 2 | Requested function | Input | Binary(4) |
| 3 | Send buffer | Input | Char(*) |
| 4 | Length of send buffer | Input | Binary(4) |
| 5 | Receive buffer | Output | Char(*) |
| 6 | Length of receive buffer | Input | Binary(4) |
| 7 | Command format | Input | Char(8) |
| 8 | Command data | Input | Char(*) |
| 9 | Length of command data | Input | Binary(4) |
| 10 | Error code | I/O | Char(*) |

Default Public Authority: *EXCLUDE

Threadsafe: Conditional; see Usage Notes.

The Control Device (QTACTLDV) API provides a direct command interface to a device. The caller of this API can issue any command directly to a device and transfer data to or from the device.

**Note:** For tape devices, the Retrieve Device Capabilities (QTARDCAP) API can be used to determine if the tape device supports the QTACTLDV API. Other kinds of devices currently do not support this API. This API can be used for a tape device within a media library if the device is deallocated from the library.

**Note:** Incorrect use of this API can cause damage to data saved on tape media or can interfere with I/O processor function.

## Authorities and Locks

*API Public Authority*

*EXCLUDE

*Special Authority*

*SERVICE

*Device Description Authority*

*CHANGE

*Device Description Lock*

*EXCLRD

## Required Parameter Group

**Device name**

INPUT; CHAR(10)

The device description to which the request is sent.

**Requested function**

INPUT; BINARY(4)

The function to perform. The possible values are:

*1* Open a connection to the device. This function must be performed before any commands can be sent to the device. The device must be varied on before this function is performed. No other job can use the device while the connection is open.

*2* Send a command to the device. When running in a multithreaded environment, a command sent by a thread must be complete before another command can be sent by another thread.

*3* Close the connection to the device. This function must be performed after the user has completed sending commands to the device. No other job can use the device until the connection is closed.

**Send buffer**

INPUT; CHAR(*)

A buffer containing data to send to the device when a data transfer command is sent. No support is provided to send and receive data on the same command.

This parameter is ignored if the length of the send buffer is 0.

**Length of send buffer**

INPUT; BINARY(4)

The length of the send buffer.

This parameter must be 0 for the open connection and close connection functions.

**Receive buffer**

OUTPUT; CHAR(*)

A buffer to store data received from the device after a data transfer command is sent. No support is provided to send and receive data on the same command.

This parameter is ignored if the length of the receive buffer is 0.

**Length of receive buffer**

INPUT; BINARY(4)

The length of the receive buffer.

This parameter must be 0 for the open connection and close connection functions.

**Command format**

INPUT; CHAR(8)

The format of the command data. The following format is supported.

*CTLD0100*

Issue command to a tape device.

**Note:** The connection must be opened using the open function before a command can be issued to the device.

See CTLD0100 Format for more information on the command data format.

**Command data**

INPUT; CHAR(*)

The variable that contains the command data.

This parameter is ignored if the length of command data is set to 0.

**Length of command data**

INPUT; BINARY(4)

The length of the command data to be sent to the device. The command data must be 0, or a minimum of 32 bytes long and a maximum of 56 bytes long.

This parameter must be 0 for the open connection and close connection functions.

**Error code**

I/O; CHAR(*)

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

## CTLD0100 Format

The following table shows the command information that is required for the CTLD0100 format. For more details about the fields in the following table, see Field Descriptions.

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | BINARY(4) | Data transfer direction |
| 4 | 4 | BINARY(4) | Requested transfer length |
| 8 | 8 | BINARY(4) | Ignore length errors |
| 12 | C | BINARY(4) | Command timeout value |
| 16 | 10 | BINARY(4) | Type of command |
| 20 | 14 | BINARY(4) | Offset to command string |
| 24 | 18 | BINARY(4) | Length of command string |
| 28 | 1C | BINARY(4) | Reserved |
| | | CHAR(*) | Command string |

## Field Descriptions

**Command string.** The command string to send to the device. See the device specifications to determine what command strings are supported by the device.

**Command timeout value.** The time, in seconds, to wait for the command to complete. Valid values are 1 through 7200.

**Data transfer direction.** The direction of any data transfer associated with the command. The possible values are:

| | |
|---|---|
| *0* | No data transfer. |
| *1* | Receive data from the device. |
| *2* | Send data to the device. |

**Ignore length errors.** The possible values are:

| | |
|---|---|
| *0* | Report length errors. |
| *1* | Ignore length errors. |

**Length of command string.** The length of the command string to send to the device. Valid values are 0 through 24.

**Offset to command string.** The offset from the start of the command data, in bytes, to the start of the command string. Valid values are 32 and greater.

**Requested transfer length.** The expected length of the data to be transferred by the command. The requested transfer length must be less than or equal to the length of the buffer parameter that will be used to send or receive the data.

**Note:** This field must be 0 for commands with no data transfer.

**Reserved.** An ignored field. This value must be set to 0.

**Type of Command.** The type of command. The possible values are:

| | |
|---|---|
| *0* | Small Computer System Interface (SCSI) command. |
| *1* | Reset the device. |

## Error Messages

For descriptions of the reason codes in CPF67C8, see Reason Codes.

| Message ID | Error Message Text |
|---|---|
| CPF222E E | &1 special authority is required. |
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF3C1D E | Length specified in parameter &1 not valid. |
| CPF3C21 E | Format name &1 is not valid. |

| | |
|---|---|
| CPF3C39 E | Value for reserved field not valid. |
| CPF3C3C E | Value for parameter &1 not valid. |
| CPF3C4C E | Value not valid for field &1. |
| CPF3C90 E | Literal value cannot be changed. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF3CF2 E | Error(s) occurred during running of &1 API. |
| CPF6708 E | Command ended due to error. |
| CPF67C8 E | Command failed for device &1. Reason code &2. |
| CPF9814 E | Device &1 not found. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

## Reason Codes

This topic contains the description of the reason codes returned in message CPF67C8.

**Note:** For command timeouts, I/O processor errors, system bus errors, and device bus errors, a reset operation might occur on the device bus. Other devices attached to the same bus may be affected by this reset operation.

Possible reason code values are:

| | |
|---|---|
| *'010300'x* | Command length not valid: The command length was too long for existing device interface specifications. |
| *'010700'x* | Data length not valid: The IOP does not support the size of the data transfer. Use the Retrieve Device Capabilities (QTARDCAP) API to determine the maximum block size supported. |
| *'020900'x* | Insufficient data: The data transfer buffer is not large enough. |
| *'02C0yy'x* | Device detected error: The tape device reported an error condition. For an SCSI type of command, yy is set to the value of the completion status. |
| *'02C100'x* | Selection timeout: The tape device did not respond to the command. Check device power and cables and retry the command. If the problem persists, contact your hardware service provider. |
| *'02C200'x* | I/O processor length error: Length error on the data transfer. The ignore length errors field was not set in the command data. |
| *'02C300'x* | I/O processor error: The I/O processor card detected an internal hardware failure. Contact your hardware service provider. |
| *'02C400'x* | Command timed out: The tape device did not complete the requested command within the specified time. Correct the command timeout value and retry the command. If the problem persists, contact your hardware service provider. |
| *'02C500'x* | System bus error: The host internal system bus failed. Contact your hardware service provider. |
| *'02C600'x* | Device bus error: The I/O processor detected a failure in the device interface. Check the device power and cables and retry the command. If the problem persists, contact your hardware service provider. |

| | |
|---|---|
| *'100000'x* | Open failure: A connection could not be opened to the device. The device may not be varied on or is being used by another job. |
| *'100001'x* | Open failure: A connection could not be opened to the device. The device does not support the QTACTLDV API. |
| *'100002'x* | Open failure: A connection could not be opened to the device. The device is in a failed state. |
| *'200000'x* | Close failure: The connection to the device could not be closed. The device may not be varied on or is being used by another job. |
| *'200002'x* | Close failure: The connection to the device could not be closed. The device is in a failed state. |
| *'300000'x* | Device not valid: The device specified is not a tape device. |
| *'300001'x* | Resource not valid: The resource name associated with the specified device is not valid or does not exist. |
| *'400000'x* | Connection not open: The command could not be completed because there is not an open connection. |

## Usage Notes

When running in a multithreaded environment, a command sent by a thread to a device must be complete before a command can be sent by another thread to the same device.

## Usage Example

See Using the QTACTLDV API in API examples for an example of how to use the QTACTLDV API.

---

API introduced: V4R4

---

# Convert Date and Time Format (QWCCVTDT) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Input format | Input | Char(10) |
| 2 | Input variable | Input | Char(*) |
| 3 | Output format | Input | Char(10) |
| 4 | Output variable | Output | Char(*) |
| 5 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: Yes

The Convert Date and Time Format (QWCCVTDT) API converts date and time values from one format to another format. The QWCCVTDT API lets you:

- Convert a time-stamp (*DTS, for system time-stamp) value to character format
- Convert a character date-time value to time-stamp format
- Convert a date from one character format to another
- Retrieve the current machine clock time and return it in the format you specify

For the ILE CEE date and time APIs, see Date and Time APIs.

## Required Parameter Group

**Input format**

INPUT; CHAR(10)

The format of the data you give QWCCVTDT to convert. Valid values are:

| | |
|---|---|
| **CURRENT* | Current machine clock time. |
| **DTS* | System time-stamp. When you convert a character date-time value to *DTS and back to character format, there is a rounding error of plus or minus 2 milliseconds. |
| **JOB* | The format given in the DATFMT job attribute. |
| **SYSVAL* | The format given in the QDATFMT system value. |
| **YMD* | YYMMDD (year, month, day) format. |
| **YYMD* | YYYYMMDD (4-digit year, month, day) format. |
| **MDY* | MMDDYY (month, day, year) format. |

| | |
|---|---|
| **MDYY* | MMDDYYYY (month, day, 4-digit year) format. |
| **DMY* | DDMMYY (day, month, year) format. |
| **DMYY* | DDMMYYYY (day, month, 4-digit year) format. |
| **JUL* | Julian format (YYDDD (year, day of year)). |
| **LONGJUL* | Long Julian format (YYYYDDD (4-digit year, day of year)). |

All date values range only from August 23, 1928, 12:03:06.315 to May 10, 2071, 11:56:53.684. Converting a date outside this range to any of these formats results in a date within this range.

You can convert any format except *CURRENT to the same format without receiving an error. When you convert a format to the same format, the input variable is copied into the output variable without validation.

When you convert one character date format (that is, anything other than *CURRENT, the current machine-clock time, or *DTS, the system time-stamp) to another character date format, the date information is validated and converted. However, the time and milliseconds portions of the input variable are copied into the output variable without validation. Also, if both the input format and the output format specify a century digit and a 2-digit year (for example, converting *YMD to *MDY), the century digit of the input variable is copied to the output variable without validation.

**Input variable**

INPUT; CHAR(*)

The data to be converted. See Input Variable Format to determine the input variable.

**Output format**

INPUT; CHAR(10)

The format to convert the data to. Valid values are:

| | |
|---|---|
| **DTS* | System time-stamp. For additional information, see the description of the *DTS value under the input format parameter. |
| **JOB* | The format given in the DATFMT job attribute |
| **SYSVAL* | The format given in the QDATFMT system value |
| **YMD* | YYMMDD format |
| **YYMD* | YYYYMMDD format |
| **MDY* | MMDDYY format |
| **MDYY* | MMDDYYYY format |
| **DMY* | DDMMYY format |
| **DMYY* | DDMMYYYY format |
| **JUL* | Julian format (YYDDD) |
| **LONGJUL* | Long Julian format (YYYYDDD) |
| **DOS* | DOSGetDateTime format. The *DOS value can be specified only when *CURRENT or *DTS is specified for the input format parameter. |

**Output variable**

OUTPUT; CHAR(*)

The converted data. If the output format is *DTS, the first 8 characters of this parameter are used. If the output format is *DOS, the first 11 characters of this parameter are used. For details, see DOSGetDateTime Value Structure. If the output format is *YYMD, *MDYY, *DMYY or *LONGJUL, the first 17 characters of the output variable are used. For details, see 17-Byte Character Date and Time Value Structure. If the output format is one of the other character formats (that is, anything other than *DTS, *DOS, *YYMD, *MDYY, *DMYY or *LONGJUL) the first 16 characters of the output variable are used. For details, see 16-Byte Character Date and Time Value Structure.

**Error code**

I/O; CHAR(*)

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

## Input Variable Format

This table shows the format used for the input variable parameter.

| **Input Format** | **Input Variable** |
|---|---|
| *CURRENT | Parameter is ignored. |
| *DTS | System time-stamp. The first 8 characters are used. |
| *YYMD, *MDYY, *DMYY, *LONGJUL | The first 17 characters are used. See 17-Byte Character Date and Time Value Structure. |
| All other character formats | The first 16 characters are used. See 16-Byte Character Date and Time Value Structure. |

## 16-Byte Character Date and Time Value Structure

This table shows the structure used for the input and output variables when the format is *JOB, *SYSVAL, *YMD, *MDY, *DMY and *JUL.

| **Offset** | **Description** |
|---|---|
| 0 | Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*. |
| 1-6 | Date, left-justified. This value cannot be all blanks or all zeros. Left-justify Julian dates, using blanks to fill the space. |
| 7-12 | Time, in HHMMSS (hours, minutes, seconds) format. |
| 13-15 | Milliseconds. This value cannot be blanks. |

## 17-Byte Character Date and Time Value Structure

This table shows the structure used for the input and output variables when the format is *YYMD, *MDYY, *DMYY and *LONGJUL.

| Offset | Description |
|---|---|
| 0-7 | Date, left-justified. This value cannot be all blanks or all zeros. Left-justify Julian dates, using blanks to fill the space. |
| 8-13 | Time, in HHMMSS (hours, minutes, seconds) format. |
| 14-16 | Milliseconds. This value cannot be blanks. |

## DOSGetDateTime Value Structure

This table shows the structure used for the output variables.

| Offset | Description |
|---|---|
| 0 | Hours (0-23)[1] |
| 1 | Minutes (0-59)[1] |
| 2 | Seconds (0-59)[1] |
| 3 | Hundredths of seconds (0-99)[1] |
| 4 | Day (1-31)[1] |
| 5 | Month (1-12)[1] |
| 6-7 | Year (for example, 1995)[2] |
| 8-9 | Time zone. This is the negative value of the system value QUTCOFFSET converted into minutes.[2] |
| 10 | Day of the week. Sunday is 0. (0-6)[1] |
| **Notes:**<br>[1] A 1-byte integer.<br>[2] A 2-byte integer. | |

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF1060 E | Date not valid. |
| CPF1061 E | Time not valid. |
| CPF1848 E | Century digit &1 not valid |

| | |
|---|---|
| CPF1849 E | Milliseconds value &1 not valid |
| CPF1850 E | Format &1 not valid |
| CPF24B4 E | Severe error while addressing parameter list. |
| 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

---

# Qp0zCvtToMITime()-Convert Timeval Structure to _MI_Time

Syntax:

```
#include <qp0z1170.h>

int Qp0zCvtToMITime (_MI_Time to,
                     const struct timeval *from,
                     int option);
```

Service Program Name: QP0ZCPA

Default Public Authority: *USE

Threadsafe: Yes

The **Qp0zCvtToMITime()** function converts a UNIX-type timestamp (or a timestamp offset), represented by a timeval structure, to a corresponding _MI_Time data type. The system value QUTCOFFSET and epoch-1970 are optionally taken into account by this conversion. Only timestamps or timestamp offsets in the following ranges can be converted:

- Timestamps: later than or equal to 1 January 1970, 00:00:00 UTC (epoch-1970) and less than 19 January 2038, 03:14:08 UTC.
- Timestamp offsets: greater than or equal to 0 and less than 2,147,483,648 seconds.

**Note:** This function uses a header (include) file from the library QSYSINC, which is optionally installable. Make sure QSYSINC is installed on your system before using this function. See Header Files for UNIX-Type Functions) for the file and member name of each header file.

## Parameters

**to**

(Output) The _MI_Time data type to contain the converted timestamp (or timestamp offset).

**from**

(Input) The address of the timeval structure to be converted.

**option**

(Input) The conversion option.

The **option** parameter must be one of the following constants:

**QP0Z_CVTTIME_TO_OFFSET**

Do the conversion as a timestamp offset, not factoring in UTC offset or epoch-1970.

**QP0Z_CVTTIME_TO_TIMESTAMP**
Do the conversion as a timestamp, factoring in the UTC offset and epoch-1970.

**QP0Z_CVTTIME_FACTOR_EPOCH_ONLY**
Do the conversion as a timestamp, but factor in epoch-1970 only.

**QP0Z_CVTTIME_FACTOR_UTCOFFSET_ONLY**
Do the conversion as a timestamp, but factor in the UTC offset only.

## Authorities and Locks

None.

## Return Value

*0* **Qp0zCvtToMITime()** was successful. The value referenced by the **to** parameter is the converted timestamp (or timestamp offset).

*-1* **Qp0zCvtToMITime()** was not successful. The *errno* variable is set to indicate the error.

## Error Conditions

If **Qp0zCvtToMITime()** is not successful, *errno* usually indicates one of the following errors. Under some conditions, *errno* could indicate an error other than those listed here.

*[EINVAL]*

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

*[EFAULT]*

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

*[ERANGE]*

A range error occurred.

The value of an argument is too small, or a result too large.

*[EUNKNOWN]*

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

## Error Messages

None.

## Usage Notes

1. **Qp0zCvtToMITime()**, when called with **option** equal to QP0Z_CVTTIME_TO_OFFSET, will convert the number of seconds and microseconds given in the **from** parameter to an equivalent machine timestamp offset, similar to what the mitime() API does.
2. **Qp0zCvtToMITime()**, when called with **option** equal to QP0Z_CVTTIME_TO_TIMESTAMP, will convert the number of seconds and microseconds given in the **from** parameter to an equivalent machine timestamp.

## Related Information

- The <**qp0z1170.h**> file (see Header Files for UNIX-Type Functions)
- Qp0zCvtToTimeval() - Convert_MI_Time to Timeval Structure

## Example

The following example converts a timestamp:

```
#include <qp0z1170.h>
#include <stdio.h>
#include <errno.h>

int main(int argc, char *argv[])
{
    _MI_Time mt;
    struct timeval tv;
    int rc;

    tv.tv_sec=867422292;
    tv.tv_usec=52992;

    printf("timeval timestamp: %u.%06u\n",
            tv.tv_sec, tv.tv_usec);
    rc = Qp0zCvtToMITime(mt, &tv,
                                 QP0Z_CVTTIME_TO_TIMESTAMP);

    if(rc==0) {
        printf("mi timestamp: %08X%08X\n",
                *((unsigned *)&mt[0]),
                *((unsigned *)&mt[4]));
    }
    else {
        printf("Qp0zCvtToMITime() failed, errno = %d\n",
                errno);
        return -1;
    }

    return 0;
}
```

**Example Output:**

```
timeval timestamp: 867422292.052992
mi timestamp: 7B7E9425EAC00000
```

---

API introduced: V4R2

---

# Qp0zCvtToTimeval()-Convert _MI_Time to Timeval Structure

Syntax:

```
    #include <qp0z1170.h>

    int Qp0zCvtToTimeval (struct timeval *to,
                          const _MI_Time from,
                          int option);
```

Service Program Name: QP0ZCPA

Default Public Authority: *USE

Threadsafe: Yes

The **Qp0zCvtToTimeval()** function converts a machine timestamp (or a machine timestamp offset), represented by an _MI_Time data type, to a corresponding structure timeval value. The system value QUTCOFFSET and epoch-1970 are optionally taken into account by this conversion. Only timestamps or timestamp offsets in the following ranges can be converted:

- Timestamps: later than or equal to 1 January 1970, 00:00:00 UTC (epoch-1970) and less than 19 January 2038, 03:14:08 UTC.
- Timestamp offsets: greater than or equal to 0 and less than 2,147,483,648 seconds.

**Note:** This function uses a header (include) file from the library QSYSINC, which is optionally installable. Make sure QSYSINC is installed on your system before using this function. See Header Files for UNIX-Type Functions) for the file and member name of each header file.

## Parameters

**to**

(Output) The address of the timeval structure to contain the converted timestamp (or timestamp offset).

**from**

(Input) The _MI_Time data type to be converted.

**option**

(Input) The conversion option.

The **option** parameter must be one of the following constants:

**QP0Z_CVTTIME_TO_OFFSET**

Do the conversion as a timestamp offset, not factoring in UTC offset or epoch-1970.

**QP0Z_CVTTIME_TO_TIMESTAMP**
Do the conversion as a timestamp, factoring in the UTC offset and epoch-1970.

**QP0Z_CVTTIME_FACTOR_EPOCH_ONLY**
Do the conversion as a timestamp, but factor in epoch-1970 only.

**QP0Z_CVTTIME_FACTOR_UTCOFFSET_ONLY**
Do the conversion as a timestamp, but factor in the UTC offset only.

## Authorities and Locks

None.

## Return Value

*0* **Qp0zCvtToTimeval()** was successful. The value referenced by the **to** parameter is the converted timestamp (or timestamp offset).

*-1* **Qp0zCvtToTimeval()** was not successful. The *errno* variable is set to indicate the error.

## Error Conditions

If **Qp0zCvtToTimeval()** is not successful, *errno* usually indicates one of the following errors. Under some conditions, *errno* could indicate an error other than those listed here.

*[EINVAL]*

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

*[EFAULT]*

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

*[ERANGE]*

A range error occurred.

The value of an argument is too small, or a result too large.

*[EUNKNOWN]*

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

## Error Messages

None.

## Usage Notes

1. **Qp0zCvtToTimeval()**, when called with **option** equal to QP0Z_CVTTIME_TO_OFFSET, will convert the machine timestamp offset given in the **from** parameter to an equivalent number of seconds and microseconds. This could be used to calculate a time delay.
2. **Qp0zCvtToTimeval()**, when called with **option** equal to QP0Z_CVTTIME_TO_TIMESTAMP, will convert the machine timestamp given in the **from** parameter to an equivalent number of seconds and microseconds. This could be used as a UNIX-type timestamp.

## Related Information

- The <**qp0z1170.h**> file (see Header Files for UNIX-Type Functions)
- Qp0zCvtToMITime() - Convert Timeval Structure to_MI_Time

## Example

The following example converts a timestamp:

```
#include <qp0z1170.h>
#include <mimchint.h>
#include <stdio.h>
#include <errno.h>

int main(int argc, char *argv[])
{
    _MI_Time mt;
    struct timeval tv;
    int rc;

    mattod(mt);

    printf("mi timestamp: %08X%08X\n",
            *((unsigned *)&mt[0]),
            *((unsigned *)&mt[4]));
    rc = Qp0zCvtToTimeval(&tv, mt, QP0Z_CVTTIME_TO_TIMESTAMP);

    if(rc==0) {
        printf("timeval timestamp: %u.%06u\n",
                tv.tv_sec, tv.tv_usec);
    }
    else {
        printf("Qp0zCvtToTimeval() failed, errno = %d\n",
                errno);
        return -1;
    }

    return 0;
}
```

**Example Output:**

```
mi timestamp: 7B7E9425EAC00000
timeval timestamp: 867422292.052992
```

---

API introduced: V4R2

---

# Generate Pseudorandom Numbers (Qc3GenPRNs) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | PRN data | Output | Char(*) |
| 2 | PRN data length | Input | Binary(4) |
| 3 | PRN type | Input | Char(1) |
| 4 | PRN Parity | Input | Char(1) |
| 5 | Error Code | I/O | Char(*) |

Service Program Name: QC3PRNG

Default Public Authority: *USE

Threadsafe: Yes

The Generate Pseudorandom Numbers (Qc3GenPRNs) API generates a pseudorandom binary stream.

The pseudorandom number generator is composed of two parts: pseudorandom number generation and seed management. Pseudorandom number generation is performed using the FIPS 186-1 algorithm. Cryptographically-secure pseudorandom numbers rely on good seed. The FIPS 186-1 key and seed values are obtained from the system seed digest. The server automatically generates seed using data collected from system information or by using the random number generator function on a cryptographic coprocessor, such as a 4758, if one is available. System-generated seed can never be truly unpredictable. If a cryptographic coprocessor is not available, you can use the Add Seed for PRNG (Qc3AddPRNGSeed) API to add your own random seed to the system seed digest. This should be done as soon as possible any time the Licensed Internal Code is installed.

## Authorities and Locks

None.

## Required Parameter Group

**PRN data**

OUTPUT; CHAR(*)

The generated pseudorandom binary stream.

**PRN data length**

INPUT; BINARY(4)

The number of pseudorandom number bytes to return in the PRN data parameter. If 0 is specified, no pseudorandom numbers are returned.

**PRN type**

INPUT; CHAR(1)

The API can generate a real pseudorandom binary stream or a test binary stream.

The FIPS 186-1 algorithm obtains the inital key and seed values from the system seed digest when generating a real pseudorandom binary stream. When generating a test binary stream, the algorithm uses preset values for the key and seed. Valid values are:

| | |
|---|---|
| *0* | Generate real pseudorandom numbers. |
| *1* | Generate test pseudorandom numbers. |

**PRN Parity**

INPUT; CHAR(1)

The API sets each byte of the pseudorandom number binary stream to the specified parity by altering the low order bit in each byte as necessary. Valid values are:

| | |
|---|---|
| *0* | Do not set parity. |
| *1* | Set each byte to odd parity. |
| *2* | Set each byte to even parity. |

**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 |
|---|---|
| CPF3C19 E | Error occurred with receiver variable specified. |
| CPF3CF1 E | Error code parameter not valid. |
| CPFBAF1 E | PRN type not valid. |
| CPFBAF2 E | Parity not valid. |
| CPFBAF3 E | The system seed digest is not ready. |

---

API introduced: V5R1

---

# Remove All Bookmarks from a Course (QEARMVBM) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Course ID | Input | Char(10) |
| 2 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Remove All Bookmarks from a Course (QEARMVBM) API removes all bookmarks from a Tutorial System Support course. This API provides support similar to option 9 (Remove bookmarks) on the Work with Courses display within the Start Education (STREDU) command.

## Authority

The user must be an education administrator and have one of the following authorities:

*Authority*

*ALLOBJ or *SECADM

## Required Parameter Group

**Course ID**

INPUT; CHAR(10)

The ID of the course that is to have all bookmarks 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 |
|---|---|
| CPF1D50 E | Not authorized to remove bookmarks. |
| CPF1D51 E | Not all bookmarks removed. |

| | |
|---|---|
| CPF1D52 E | Course not found. |
| 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

---

Top | Miscellaneous APIs | APIs by category

# Retrieve Data (QPARTVDA) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Receiver variable | Output | Char(*) |
| 2 | Length of receiver variable | Input | Binary(4) |
| 3 | Actual length of user data | Output | Binary(4) |
| 4 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Retrieve Data (QPARTVDA) API retrieves up to 1KB of user data, which was passed to this system with the Start Pass-through (QPASTRPT) API.

## Required Parameter Group

**Receiver variable**

OUTPUT; CHAR(*)

User data associated with a pass-through session. You can specify the size of the area to be smaller than the data sent by the source system as long as you specify the length parameter correctly. As a result, the API returns only the data 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.

**Actual length of user data**

OUTPUT; BINARY(4)

The actual length of user data associated with this pass-through session. If this value is greater than the length of receiver variable parameter, then truncation occurred.

**Error code**

I/O; CHAR(*)

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

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF3C19 E | Error occurred with receiver variable specified. |
| CPF3C24 E | Length of the receiver variable is not valid. |
| CPF3C90 E | Literal value cannot be changed. |
| CPF8942 E | Command or API call not allowed on source system. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V3R6

Top | Miscellaneous APIs | APIs by category

# Start Pass-Through (QPASTRPT) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Pass-through information | Input | Char(*) |
| 2 | Length of pass-through information | Input | Binary(4) |
| 3 | Format name | Input | Char(8) |
| 4 | Data | Input | Char(*) |
| 5 | Length of data | Input | Binary(4) |
| 6 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Start Pass-Through (QPASTRPT) API starts a 5250 pass-through session and optionally passes up to 1KB of user data from the source system to the target system. This data can be accessed on the target system with the Retrieve Data (QPARTVDA) API.

## Authorities and Locks

*APPC Device on Source System*

*CHANGE

*APPC Device on Target System*

*CHANGE

*Virtual Controller on Target System*

*USE

*Virtual Device on Target System*

*CHANGE

*Program Specified in QRMTSIGN System Value on Target System*

*USE

## Required Parameter Group

**Pass-through information**

INPUT; CHAR(*)

Information associated with establishing the 5250 pass-through session.

**Length of pass-through information**

INPUT; BINARY(4)

The length, in bytes, of the pass-through information parameter. This value must be greater than or equal to 8 and less than or equal to 580.

**Format name**

INPUT; CHAR(8)

The format of the pass-through information. The supported format names are:

| | |
|---|---|
| *PAST0100* | Pass-through with up to 10-byte password |
| *PAST0200* | Pass-through with up to 128-byte password |

See PAST0100 Format and PAST0200 Format for details.

**Data**

INPUT; CHAR(*)

User-defined data to be passed to the target system. The format of this data is not defined by the API, and is sent to the target system as is.

**Length of data**

INPUT; BINARY(4)

The length of the data parameter.

**Error code**

I/O; CHAR(*)

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

## PAST0100 Format

The following table is the layout of the pass-through information for format PAST0100, which controls how the pass-through session is established for a pass-through with up to a 10-byte password.

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | CHAR(8) | Remote location name |
| 8 | 8 | CHAR(10) | Virtual controller |
| 18 | 12 | CHAR(8) | Mode name |
| 26 | 1A | CHAR(8) | Local location name |
| 34 | 22 | CHAR(8) | Remote network ID |
| 42 | 2C | CHAR(10) | System request program name |
| 52 | 34 | CHAR(10) | System request library name |
| 62 | 3E | CHAR(10) | Remote user ID |
| 72 | 48 | CHAR(10) | Remote password |
| 82 | 52 | CHAR(10) | Initial program |
| 92 | 5C | CHAR(10) | Initial menu |
| 102 | 66 | CHAR(10) | Current library |

| | | | |
|---|---|---|---|
| 112 | 70 | CHAR(1) | Display option |
| 113 | 71 | CHAR(3) | Reserved |
| 116 | 74 | BINARY(4) | Offset to virtual devices |
| 120 | 78 | BINARY(4) | Number of virtual devices |
| | | CHAR(*) | Array of virtual devices |

## »PAST0200 Format

The following table is the layout of the pass-through information for format PAST0200, which controls how the pass-through session is established for a pass-through with up to a 128-byte password.

| **Offset** | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | CHAR(8) | Remote location name |
| 8 | 8 | CHAR(10) | Virtual controller |
| 18 | 12 | CHAR(8) | Mode name |
| 26 | 1A | CHAR(8) | Local location name |
| 34 | 22 | CHAR(8) | Remote network ID |
| 42 | 2C | CHAR(10) | System request program name |
| 52 | 34 | CHAR(10) | System request library name |
| 62 | 3E | CHAR(10) | Remote user ID |
| 72 | 48 | CHAR(10) | Reserved |
| 82 | 52 | CHAR(10) | Initial program |
| 92 | 5C | CHAR(10) | Initial menu |
| 102 | 66 | CHAR(10) | Current library |
| 112 | 70 | CHAR(1) | Display option |
| 113 | 71 | CHAR(3) | Reserved |
| 116 | 74 | BINARY(4) | Offset to virtual devices |
| 120 | 78 | BINARY(4) | Number of virtual devices |
| 124 | 7C | BINARY(4) | Offset to remote password |
| 128 | 80 | BINARY(4) | Phrase length of remote password |
| | | CHAR(*) | Remote password |
| | | CHAR(*) | Array of virtual devices« |

## Field Descriptions

**Array of virtual devices.** An array of 0 through 32 virtual devices on the target system. A device on the target system is selected from this list based on a comparison of device type and model.

**Current library.** The library to be the current library on the target system. The special value *RMTUSRPRF can be used to indicate that the current library found in the remote user profile should be used.

**Display option.** Whether the pass-through and associated status messages appear. Special values follow:

| | |
|---|---|
| *0* | Do not display pass-through information. |
| *1* | Display pass-through information. |

**Initial menu.** The menu that is initially shown at the target system. This runs after the initial program. Special values follow:

| | |
|---|---|
| **RMTUSRPRF* | Show the initial menu as specified by the remote user profile. |
| **SIGNOFF* | Sign off the target system after running the initial program. |

**Initial program.** The program that is called immediately after sign-on to the target system. Special values follow:

| | |
|---|---|
| **RMTUSRPRF* | Call the initial program as specified by the remote user profile. |
| **NONE* | There is no initial program to call. |

**Local location name.** The name by which the local iSeries server is known to other devices in the network. Special values follow:

| | |
|---|---|
| **LOC* | The local location name is chosen by the system. |
| **NETATR* | The local location name, a system network attribute, is used. |

**Mode name.** The mode to be used. The special value *NETATR can be used to indicate that the system network attribute mode should be used.

**Number of virtual devices.** The number of virtual devices in the array of virtual devices. If the virtual controller field is not *NONE, this field must be set to 0.

»**Offset to remote password**. The offset from the beginning of the format to the start of the remote password. The phrase length of the remote password field must be a valid value.«

**Offset to virtual devices.** The offset from the beginning of the format to the start of the array of virtual devices. If the virtual controller field is not *NONE, this field must be set to 0.

»**Phrase length of remote password**. The length, in bytes, of the remote password. This value must be greater than 0 and less than or equal to 128.«

**Remote location name.** The name of the location that is the target of the pass-through session.

**Remote network ID.** The network ID of the network where the remote location resides. Special values follow:

| | |
|---|---|
| **LOC* | Any remote location name will be used. |
| **NETATR* | The local network ID, a system network attribute, is used. |
| **NONE* | The remote system does not support network IDs. |

**Remote password.** The password being sent to the target system. The special value allowed is *NONE. If a profile is specified for the remote user ID field and password security is active on the target system, *NONE is not allowed.

**Remote user ID.** The user profile for automatic sign-on to the target system. Special values follow:

| | |
|---|---|
| **NONE* | No user ID is passed to the target system; automatic sign-on is not used. |
| **CURRENT* | The user ID that is active in the current job is passed to the remote system. |

**Reserved.** An ignored field. This field must be blanks.

**System request library name.** The library in which the system request program can be found. Special values are *LIBL and *CURLIB. If the system request program is *SRQMNU, this field must be set to blanks.

**System request program name.** The program that is to be called on the source system when system request option 10 is selected. The special value *SRQMNU causes the system-supplied system request menu to be displayed.

**Virtual controller.** The name of the virtual controller on the target system that is used to do pass-through. If you specify a virtual controller, one of the virtual display devices attached to it is selected for the pass-through job. This entry is mutually exclusive of the array of virtual devices, and must be *NONE if the number of virtual devices field is not 0. The default is *NONE.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF2702 E | Device description &1 not found. |
| CPF2703 E | Controller description &1 not found. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF3C1D E | Length specified in parameter &1 not valid. |
| CPF3C21 E | Format name &1 is not valid. |
| CPF3C90 E | Literal value cannot be changed. |
| CPF5383 E | Mode &7 specified for divice &4 not valid. |
| CPF5536 E | System cannot automatically select an APPC device description for the remote location. |
| CPF5546 E | Class-of-service for device &4 not valid. |
| CPF8901 E | Virtual device &1 not varied on. |
| CPF8902 E | Virtual device &1 not available. |
| CPF8904 E | Pass-through request not accepted. |
| CPF8905 E | Pass-through not allowed on this system. |
| CPF8906 E | Error during session initialization. Reason code &1. |

| | |
|---|---|
| CPF8907 E | Communications failure for device &1. |
| CPF8908 E | Controller &1 not varied on. |
| CPF8911 E | Communications failure. Session was not started. |
| CPF8912 E | Pass-through session ended. Reason code &1. |
| CPF8913 E | Pass-through ended abnormally. |
| CPF8916 E | Cannot select virtual device &1 at system &2. |
| CPF8918 E | Job canceled at system &1. |
| CPF8919 E | Device &1 not accessed by system &2. |
| CPF8920 E | Pass-through failed. &1 must be varied off and on. |
| CPF8921 E | APPC failure. Failure code is &3. |
| CPF8922 E | Negative response from device &1 at system &2. |
| CPF8935 E | Pass-through not allowed to system &1. |
| CPF8936 E | Pass-through failed for security reasons. |
| CPF8937 E | Automatic sign on not allowed. |
| CPF8939 E | Trying to send too much data. |
| CPF8944 E | Device &1 no longer communicating with system &2. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

API introduced: V3R6

# »Device Selection Exit Program

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Format name | Input | Char(8) |
| 2 | Device selection information | Input/Output | Char(*) |
| 3 | Return code | Output | Binary(4) |

Exit point name: QIBM_QPA_DEVSEL

Exit point format name: PADS0100

QSYSINC Member Name: EPADSEL

The Device Selection exit program provides an interface to control virtual device selection and automatic creation used by the system for connection requests from clients using virtual device support. The interface allows the user to write an exit program to specify the naming conventions used for automatically created virtual devices and virtual controllers and to specify the automatic creation limit to be used for the specific request.

The exit program can:

1. Reject a specific name for a device connection request.
2. Specify a naming pattern to be used for the automatic creation of a virtual device. This is used only if a specific device name was not requested on the client's connection request.
3. Specify the naming pattern of the virtual controller to be used:
   - to search in an attempt to select an existing device (if a specific device name was not requested on the client's connection request) or
   - to specify a controller to which to attach the automatically created device.

   This naming pattern is also used to 'count' the number of existing devices toward the automatic creation limit.
4. Specify a second controller naming pattern to be used to 'count' the number of existing virtual devices.
5. Specify the number of devices that can exist on the virtual controllers whose naming pattern is specified.

## Required Parameter Group

**Format name**

INPUT; CHAR(8)

The format of the information provided in the Device selection information parameter. The format name is PDSC0100.

**Device selection information**

INPUT/OUTPUT; CHAR(*)

The structure containing the data that is being passed to the exit program and that is returned from the exit program.

**Return code**

OUTPUT; BINARY(4)

Whether to allow the connection request to continue. The possible values are:

| | |
|---|---|
| 0 | Do not allow connection request. |
| 1 | Allow connection request. |

If any other value is returned, the request for selection and automatic creation of a device description for the client request will be processed using the system defaults for the QAUTOVRT system value and the defaults for the device and controller naming conventions.

This parameter is initialized to 0 on the call to the exit program.

## PDSC0100 Format

For details about the fields in the following table, see Field Descriptions.

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | BINARY(4) | Size of structure |
| 4 | 4 | BINARY(4) | Function |
| 8 | 8 | BINARY(4) | Specific name requested |
| 12 | C | CHAR(10) | Name for requested device |
| 22 | 16 | CHAR(8) | Format of returned data |

## PDSR0100 Format

For details about the fields in the following table, see Field Descriptions.

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 30 | 1E | CHAR(2) | Reserved |
| 32 | 20 | BINARY(4) | Autocreation limit |
| 36 | 24 | CHAR(6) | Naming pattern for device |
| 42 | 2A | CHAR(6) | Controller naming pattern for device attachment |
| 48 | 30 | CHAR(6) | Additional controller naming pattern |

## Field Descriptions

**Additional controller naming pattern.** The naming pattern for the controllers to be used for the device automatic creation limit. When applying the check for automatic creation limit, devices attached to these controllers are also counted when determining ifr the limit is exceeded. This field is initialized to blanks before the call is made to the exit program.

**Autocreation limit.** The number to be used for the virtual device automatic creation limit for this connection request. Possible values are:

| | |
|---|---|
| 0 | Do not allow any additional virtual device descriptions to be created automatically. |
| 1-32500 | The number of devices that can be attached to the controller descriptions whose naming patterns are specified in Controller naming pattern for device attachment and Additional controller naming pattern. |
| 32767 | The special value of *NOMAX. Do not limit the automatic creation of virtual devices. |

**Controller naming pattern for device attachment.** The naming pattern for the controller to which an automatically created device is to be attached. These characters must be a valid input to the CRTCTLVWS command. If there are not six characters, the pattern is padded with zeros. If the Autocreation limit is 1-32500, the devices attached to controllers with this pattern are counted and this number is used toward the automatic creation limit. This field is initialized to blanks before the call is made to the exit program.

**Format of returned data.** The format name specified by the user exit program for the output data returned from the Device Selection exit point. The only format supported currently is PDSR0100. This field is initialized to PDSR0100 on the call to the exit program.

**Function.** The function being used by the client. Possible values are:

| | |
|---|---|
| *1* | APPC |
| *2* | TELNET |
| *3* | Virtual Terminal Manager API (VTM API) |

**Name for requested device.** The name of the device requested by the client. If Specific name requested is set to 0, this field is blank.

**Naming pattern for device.** The naming pattern to be used for device automatic creation. These characters must be a valid input to the CRTDEVDSP command. This field is checked only if the Specific name requested field is 0. This field is initialized to blanks before the call is made to the exit program.

**Reserved.** A reserved field that must be set to hexadecimal zeros.

**Size of structure.** The size of the structure containing the data being passed to and returned from the exit program.

**Specific name requested.** Whether a specific name was requested by the client. If a specific name was requested, this name will be passed to the exit program in the Name for requested device field.

| | |
|---|---|
| *0* | No specific name was requested. |
| *1* | Specific name was requested. |

## Coding Guidelines

Applications should consider the following when coding this exit program:

- The program should return an exception for the requested operation only if there has been a failure in the operation. If the program signals an escape message to the API, the system assumes there is a failure. A diagnostic message is returned to the calling program. The request for selection and automatic creation of a device description for the client request will be processed using the system defaults for the QAUTOVRT system value and the defaults for the device and controller naming conventions.

- The program must clean up any locks that it acquires.

- The program must handle all potential error conditions associated with its own operations (be fault tolerant).

- The program must avoid infinite looping conditions.

---

Exit program introduced: V5R2

---

# Process Open List APIs

The process open list APIs are used to access the data returned by the open list APIs. You can get list entries, find entry numbers in lists and in message lists, find field numbers in lists, retrieve server job information, and close lists. Some examples of these open list APIs are:

- Open List of Job Log Messages (QGYOLJBL)
- Open List of Messages (QGYOLMSG)
- Open List of Objects (QGYOLOBJ)
- Open List of Objects to be Backed Up (QEZOLBKL)
- Open List of Printers (QGYRPRTL)
- Open List of Spooled Files (QGYOLSPL)

These list APIs can improve perceived performance when creating lists. The APIs create and make available to the caller a partial listing of the total set of files, messages, or objects. This list is immediately available to be acted upon, while the remainder of the list is being created. The user does not have to wait for the entire list to be created.

The open list APIs are available only if the Host Servers option of OS/400 is installed. You can install this option by using the GO LICPGM function of OS/400. Select the Install Licensed Programs option on the Work with Licensed Programs display, and select the Host Servers option on the Install Licensed Programs display.

The process open list APIs and their functions are:

- Change Server Job (QGYCHGSJ) sets the maximum number of auxiliary server jobs allowed for a server job with the iSeries.
- Close List (QGYCLST) closes a previously opened list. Any internal storage associated with that list is freed.
- Find Entry Number in List (QGYFNDE) returns the number of the entry in a list of information for a given key value.
- Find Entry Number in Message List (QGYFNDME) returns the number of the entry in a list of message information for a given key value.
- Find Field Numbers in List (QGYFNDF) returns the number of the entries in a list of information and the value of that entry whenever the value of that field changes.
- Get List Entries (QGYGTLE) allows requests to get entries from previously opened lists on the server.
- Retrieve Server Job Information (QGYRTVSJ) returns information about auxiliary server jobs started for the current job to the system.

---

# Change Server Job (QGYCHGSJ) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Number of auxiliary server jobs allowed | Input | Binary(4) |
| 2 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Change Server Job (QGYCHGSJ) API sets the maximum number of auxiliary server jobs allowed for a server job on the iSeries server. At least one auxiliary server job is allowed; up to five auxiliary server jobs may be allowed. An **auxiliary server job** is used to do work asynchronously from the job that started the auxiliary server job. For example, the auxiliary server job is used to complete building lists of information. All auxiliary server jobs end automatically when the submitting job ends.

The Retrieve Server Job Information (QGYRTVSJ) API can be called to retrieve the number of active auxiliary server jobs, the number of auxiliary server jobs allowed, and the job names for each active auxiliary server job.

## Required Parameter Group

**Number of auxiliary server jobs allowed**

INPUT; BINARY(4)

The number of auxiliary server jobs that may be started for the current server job. If the number specified is less than the number that is currently allowed, no change will be made. No more than five auxiliary server jobs may be allowed.

**Error code**

I/O; CHAR(*)

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

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| 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. |
| GUI0113 E | Number of auxiliary server jobs, &1, not valid. |

---

API introduced: V3R6

---

# Close List (QGYCLST) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Request handle | Input | Char(4) |
| 2 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Close List (QGYCLST) API closes a previously opened list. Any internal storage associated with that list is freed. The handle specified on the call to this API is no longer valid after the call completes.

## Required Parameter Group

**Request handle**

INPUT; CHAR(4)

The handle to the list that is to be closed. The handle is generated by one of the following list APIs:

- Open List of Jog Log Messages (QGYOLJBL)
- Open List of Messages (QGYOLMSG)
- Open List of Objects (QGYOLOBJ)
- Open List of Printers (QGYRPRTL)
- Open List of Spooled Files (QGYOLSPL)

**Error code**

I/O; CHAR(*)

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

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| 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. |
| GUI0001 E | Invalid handle specified. |

---

API introduced: V3R6

---

Top | Miscellaneous APIs | APIs by category

# Find Entry Number in List (QGYFNDE) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Request handle | Input | Char(4) |
| 2 | Number of keys | Input | Binary(4) |
| 3 | Key field information | Input | Array(*) of Char(12) |
| 4 | Key field values | Input | Array(*) of Char(30) |
| 5 | Entry number | Output | Binary(4) |
| 6 | Key found | Output | Char(1) |
| 7 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Find Entry Number in List (QGYFNDE) API returns the number of the entry in a list of information for a given key value.

## Required Parameter Group

**Request handle**

INPUT; CHAR(4)

The handle of the list. This handle is generated by one of the following open list APIs:

- Open List of Objects (QGYOLOBJ)
- Open List of Printers (QGYRPRTL)
- Open List of Spooled Files (QGYOLSPL)

**Number of keys**

INPUT; BINARY(4)

The number of elements, within the key field information array, to search.

**Key field information**

INPUT; ARRAY(*) of CHAR(12)

The offset and length of the information to search.

| | |
|---|---|
| *Key field offset* | INPUT; BINARY(4)<br>The offset within the list entry to search. |
| *Key field length* | INPUT; BINARY(4)<br>The length of the field within the list entry to search. |

| | |
|---|---|
| *Reserved* | INPUT; CHAR(4)<br>An ignored field. This field must be set to hexadecimal zeros. |

**Key field values**

INPUT; ARRAY(*) of CHAR(30)

The value of the fields indicated in the key field information parameter for which to search.

**Entry number**

OUTPUT; BINARY(4)

The number of the first entry in the list in which the key is found. If the key is not found in a sorted list, the number of the entry previous to where the requested entry would have been is returned (a 1 is returned if the entry not found is the first entry in the list). If the key is not found in an unsorted list, a 1 is returned. If the list is empty, a 0 is returned.

**Key found**

OUTPUT; CHAR(1)

Whether the entry returned is for the key requested or the key was not found. The possible values are:

| | |
|---|---|
| *0* | The key was not found. The entry number returned is not associated with the key given. |
| *1* | The key was found. The entry number returned is associated with the key given. |

**Error code**

I/O; CHAR(*)

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

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| 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. |
| GUI0001 E | Invalid handle specified. |
| GUI0066 E | Number of keys must be at least &2. |

API introduced: V3R6

# Find Entry Number in Message List (QGYFNDME) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Request handle | Input | Char(4) |
| 2 | Number of keys | Input | Binary(4) |
| 3 | Key field information | Input | Array(*) of Char(12) |
| 4 | Key field values | Input | Array(*) of Char(30) |
| 5 | Entry number | Output | Binary(4) |
| 6 | Key found | Output | Char(1) |
| 7 | Message type information | Input | Char(10) |
| 8 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Find Entry Number in Message List (QGYFNDME) API returns the number of the entry in a list of messages for a given key value. This API may be used on a list generated by the following APIs:

- Open List of Job Log Messages (QGYOLJBL)
- Open List of Messages (QGYOLMSG)

## Required Parameter Group

**Request handle**

INPUT; CHAR(4)

The handle of the list, generated by one of the list messages APIs.

**Number of keys**

INPUT; BINARY(4)

The number of elements, within the key field information array, to search.

**Key field information**

INPUT; ARRAY(*) of CHAR(12)

The offset and length of information to search.

| | |
|---|---|
| *Key field offset* | BINARY(4)<br>The offset within the list entry to search. |
| *Key field length* | BINARY(4)<br>The length of the field within the list entry to search. |

| | |
|---|---|
| *Reserved* | CHAR(4)<br>An ignored field. This field must be set to hexadecimal zeros. |

**Key field values**

INPUT; ARRAY(*) of CHAR(30)

The value of the fields indicated in the key field information parameter to search for.

**Entry number**

OUTPUT; BINARY(4)

The number of the first entry in the list in which the key is found. If the key is not found in a sorted list, the number of the entry previous to where the requested entry would have been is returned (a 1 is returned if the entry not found is the first entry in the list). If the key is not found in an unsorted list, a 1 is returned. If the list is empty, a 0 is returned.

**Key found**

OUTPUT; CHAR(1)

Whether the entry returned is for the key requested or the key was not found. The possible values are:

| | |
|---|---|
| *0* | The key was not found. The entry number returned is not associated with the key given. |
| *1* | The key was found. The entry number returned is associated with the key given. |

**Message type information**

INPUT; CHAR(10)

The type of message to search for. A valid value must be specified if the messages have been grouped. The possible values are:

| | |
|---|---|
| **MNR* | The group of messages that need a reply are searched. |
| **MNNR* | The group of messages that do not need a reply are searched. |
| **SCNR* | The group of sender's copy messages that need a reply are searched. |

**Error code**

I/O; CHAR(*)

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

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |

| | |
|---|---|
| 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. |
| GUI0001 E | Invalid handle specified. |
| GUI0067 E | &1 is not valid for message type. |
| GUI0102 E | Offset of field, &1, is not valid. |

---

API introduced: V3R6

---

# Find Field Numbers in List (QGYFNDF) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Request handle | Input | Char(4) |
| 2 | Receiver variable | Output | Char(*) |
| 3 | Length of receiver variable | Input | Binary(4) |
| 4 | Format | Input | Char(8) |
| 5 | Field specification | Input | Char(*) |
| 6 | Total number returned | Output | Binary(4) |
| 7 | Record number | I/O | Binary(4) |
| 8 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Find Field Numbers in List (QGYFNDF) API returns the number of the entry in a list of information and the value of that field whenever the value of that field changes. Two types of find operations are supported.

- Generic find operations: The caller specifies which field in the record is used to cause a break.
- Formatted find operations: The format selected determines which field or fields cause a break.

## Required Parameter Group

**Request handle**

INPUT; CHAR(4)

The handle of the request. When a list API is called, a handle is returned upon successful completion. One of these handles is required as input to this API.

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

The receiver variable must be large enough to hold one array entry. For format FNDF0100 the minimum receiver length must be:

```
4 + field length rounded up to a multiple of 4
```

For format FNDF0200 the minimum receiver length must be:

```
40
```

**Format name**

INPUT; CHAR(8)

The content and format of the information returned. The possible format names are:

| | |
|---|---|
| *FNDF0100* | Generic find operation is performed.<br><br>The calling program specifies the field offset and length in the field specification parameter.<br><br>Each time the field changes in the list, a record entry is added to the receiver variable. |
| *FNDF0200* | Spooled file find operation by printer is performed.<br><br>This format can only be used against a list of spooled files with format OSPL0200 that contains the following fields:<br>❍ Device name (field key 208)<br>❍ Output queue library (field key 207)<br>❍ Output queue name (field key 206)<br><br>Each time the device name field changes in the list, a record is added to the receiver variable. In addition, when the printer assigned value is 2, then each time the output queue changes in the list a record is added to the receiver variable. |

**Field specification**

INPUT; CHAR(*)

The fields to search for a break. For the FNDF0100 format this parameter must be an array of field offsets and field lengths. The fields specified are considered one logical field for comparison when searching for changes in the fields. The values for the fields will be returned in the receiver variable concatenated in the same order they are specified in this parameter. For the FNDF0100 format, this parameter must have the following layout:

| Offset | | | |
|---|---|---|---|
| **Dec** | **Hex** | **Type** | **Field** |
| 0 | 0 | BINARY(4) | Number of fields to search on. Valid values are 1 through 3. |
| Offsets vary. These fields repeat, in the order listed, for each field to search on. | | BINARY(4) | Field offset |
| | | BINARY(4) | Field length |

For the FNDF0200 format, this parameter must have the following layout:

| Offset | | | |
|---|---|---|---|
| Dec | Hex | Type | Field |
| 0 | 0 | BINARY(4) | Number of fields to search on. This must be set to zero (0). |

**Total number returned**

OUTPUT; BINARY(4)

The total number of array entries returned in the receiver variable. This is not necessarily the total number of entries available. If the receiver variable is not large enough to hold all available entries, the record number parameter is set to the record number where the next break occurs.

**Record number**

I/O; BINARY(4)

On input, the record to begin searching for field breaks. This should be set to 1 for the initial call for a list so that searching begins with the first record. This program always assumes a field break to have occurred at the record specified by this parameter. Thus, the record specified is always returned in the receiver variable.

On output, this parameter indicates whether the information in the receiver variable is partial or complete. If the receiver variable is complete, this parameter will be set to zero. If the receiver variable contains partial information, this parameter will be set to the record number where the next field break occurs. This value can be specified as input on a subsequent call to this program to continue the search for field breaks.

**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

### Format FNDF0100

| Offset | | | |
|---|---|---|---|
| Dec | Hex | Type | Field |
| 0 | 0 | BINARY(4) | Record size |
| Offsets vary. These fields repeat, in the order listed, for each entry returned. | | BINARY(4) | Record number |
| | | CHAR(*) | Field value |
| | | CHAR(*) | Padding for boundary alignment |

### Format FNDF0200

| Offset | | Type | Field |
|---|---|---|---|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Record size |
| Offsets vary. These fields repeat, in the order listed, for each entry returned. | | BINARY(4) | Record number |
| | | CHAR(10) | Output queue name |
| | | CHAR(10) | Output queue library |
| | | CHAR(1) | Printer assigned value |
| | | CHAR(10) | Device name |
| | | CHAR(*) | Padding for boundary alignment |

## Field Descriptions

**Device name.** The name of the printer device where the spooled file is printed.

**Field value.** The value of the field where the break occurs.

This is a concatenation of the fields specified to search on. The fields are concatenated in the order they were specified in the field specification parameter. The length of this field is the sum of all field lengths specified in the field specification parameter.

**Output queue library.** The name of the library containing the output queue that the spooled file is assigned.

**Output queue name.** The name of the output queue that the spooled file is assigned.

**Padding for boundary alignment.** A reserved field.

**Printer assigned value.** The value specifying whether this spooled file is assigned to a printer. Valid values are 1 through 3.

| Printer Value | Description |
|---|---|
| 1 | Spooled file is assigned to a specific printer. The device name field contains the name of the printer. |
| 2 | Spooled file is assigned to multiple printers. The device name field is set to blanks. |
| 3 | Spooled file is not assigned to a printer. The device name field is set to blanks. |

**Record number.** The record number in the list where the break occurs.

**Record size.** The size of the record in the array shown.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF3C19 E | Error occurred with receiver variable specified. |
| CPF3C21 E | Format name &1 is 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. |
| GUI0001 E | Invalid handle specified. |
| GUI0002 E | &2 is not valid for length of receiver variable. |
| GUI0101 E | List information is not valid. Reason code &1. |
| GUI0102 E | Offset of field, &1, is not valid. |
| GUI0103 E | Invalid handle specified. Length of field, &2, is not valid. Reason code &1. |
| GUI0104 E | Record number, &1, is not valid. |
| GUI0106 E | List does not contain required fields. |
| GUI0107 E | Number of fields, &2, is not valid. Reason code &1. |

---

API introduced: V3R6

---

# Get List Entries (QGYGTLE) API

Required Parameter Group:

| | | | |
|---|---|---|---|
| 1 | Receiver variable | Output | Char(*) |
| 2 | Length of receiver variable | Input | Binary(4) |
| 3 | Request handle | Input | Char(4) |
| 4 | List information | Output | Char(80) |
| 5 | Number of records to return | Input | Binary(4) |
| 6 | Starting record | Input | Binary(4) |
| 7 | Error code | I/O | Char(*) |

Default Public Authority: *USE

Threadsafe: No

The Get List Entries (QGYGTLE) API allows requests to get entries from previously opened lists on the iSeries server. A list will exist if an initial request has already been made and the list was not closed using the Close List (QGYCLST) API.

Initial requests are made by calling the following APIs:

- Open List of Job Log Messages (QGYOLJBL)
- Open List of Messages (QGYOLMSG)
- Open List of Objects (QGYOLOBJ)
- Open List of Printers (QGYRPRTL)
- Open List of Spooled Files (QGYOLSPL)
- Open List of User Certificates (QSYOLUC)
- Open List of Validation List Entries (QSYOLVLE)
- Retrieve Objects Secured by Authorization List (QGYRATLO)

## 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.

**Request handle**

INPUT; CHAR(4)

The handle of the request. The value of this determines from which user space to retrieve the information.

**List information**

OUTPUT; CHAR(80)

Information about the list from which entries are being returned. 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 (starting with the record indicated in the starting record parameter) to take from the user space and put into the receiver variable. This value must be greater than or equal to zero.

If the value zero is specified, then only the list information is returned and no actual list entries are returned.

**Starting record**

INPUT; BINARY(4)

The entry in the list that will be the first entry to be put into the receiver variable. The value must be greater than zero or one of the special values of 0 or -1.

The special value of 0 indicates that the list information should be returned to the caller immediately. The special value 0 is only allowed when the number of records to return parameter is zero.

The special value of -1 indicates that the whole list should be built before the list information is returned to the caller.

The following table shows how the number of records to return and the starting record parameters interact with each other. The record parameter is represented by an X. The number of records to return parameter is represented by a Y.

| **Starting Record (X)** | **Number of records to return (Y=0)** | **Number of records to return (Y>0)** |
|---|---|---|
| X = 0 | Immediately return only the list information | Invalid combination. An error message is sent. |
| X = -1 | Return only the list information, but wait until the whole list is built | Wait until the whole list is built and then return the number of records requested from the end of the list. See note. |
| **Note:** If the receiver variable is not large enough to hold the number of records requested, then only those that will fit into the receiver variable will be returned, but they will always be the last ones in the list. | | |

| | | |
|---|---|---|
| X > 0 | Return only the list information, but wait until list entries have been built, as specified in the starting record parameter. | Return the number of list entries specified in the number of records to return parameter starting with the entry specified in the starting record parameter. |

**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 format of the receiver variable was specified when the list was originally created.

## Error Messages

| Message ID | Error Message Text |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF3C19 E | Error occurred with receiver variable specified. |
| 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. |
| GUI0001 E | Invalid handle specified. |
| GUI0002 E | &2 is not valid for length of receiver variable. |
| GUI0006 E | &1 is not valid for starting record number. |
| GUI0027 E | &1 is not valid for number of records to return. |
| GUI0114 E | The list cannot be completed. No server jobs are running or can be started. |
| GUI0115 E | The list has been marked in error. See the previous messages. |
| GUI0118 E | Starting record cannot be 0 when records have been requested. |

API introduced: V3R6

# Retrieve Server Job Information (QGYRTVSJ) 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(*) |

Default Public Authority: *USE

Threadsafe: No

The Retrieve Server Job Information (QGYRTVSJ) API returns information about auxiliary server jobs started for the current job on the iSeries server. This API will return the number of auxiliary server jobs and the job names for each auxiliary server job. This information can be used to:

- Retrieve information about the auxiliary server jobs by calling the Retrieve Job Information (QUSRJOBI) API.
- Change the run-time attributes of one or more auxiliary server jobs using the Change Job (CHGJOB) CL command.

An **auxiliary server job** is used to do work asynchronously from the job that started the auxiliary server job. For example, the auxiliary server job is used to complete building lists of information.

The Change Server Job (QGYCHGSJ) API can be used to change the maximum number of auxiliary server jobs that can be active at any one time.

## 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 information to be returned. You can use this format:

| | |
|---|---|
| *SJBI0100* | Basic auxiliary server job information. For details see SJBI0100 Format. |

**Error code**

I/O; CHAR(*)

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

## SJBI0100 Format

The following table describes the information returned in the receiver variable for the SJBI0100 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) | Number of active auxiliary server jobs |
| 12 | C | BINARY(4) | Number of auxiliary server jobs allowed |
| 16 | 10 | BINARY(4) | Offset to auxiliary server job information |
| 20 | 14 | BINARY(4) | Job information record size |
| Offsets vary. These fields repeat for each active auxiliary server job. | | CHAR(26) | Qualified auxiliary server job name |
| | | CHAR(16) | Internal job identifier |

## Field Descriptions

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

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

**Internal job identifier.** A value sent to other APIs to speed the process of locating the job on the system. Only OS/400 APIs use this identifier. The identifier is not valid following an initial program load (IPL). If you attempt to use it after an IPL, an exception occurs.

**Job information record size.** The length of the auxiliary server job information record.

**Number of active auxiliary server jobs.** The number of auxiliary server jobs currently running for the current job.

**Number of auxiliary server jobs allowed.** The number of auxiliary server jobs allowed to be running at one time. The values are 1 through 5.

**Offset to auxiliary server job information.** The offset from the beginning of the receiver variable to the beginning of the auxiliary server job names.

**Qualified auxiliary server job name.** The qualified job name of the auxiliary server job. The qualified job name consists of the following fields:

| | |
|---|---|
| *CHAR(10)* | Job name |
| *CHAR(10)* | User name |
| *CHAR(6)* | Job number |

## Error Messages

| **Message ID** | **Error Message Text** |
|---|---|
| CPF24B4 E | Severe error while addressing parameter list. |
| 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. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |

---

API introduced: V3R6

---

Top | Miscellaneous APIs | APIs by category