System i

Programming
Miscellaneous APIs

Version 6 Release 1

System i

Programming
Miscellaneous APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in “Notices,” on page 45.

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

Contents

Miscellaneous APIs

The miscellaneous APIs are the APIs that do not logically fall in a specific part of the i5/OS® reference information.

The miscellaneous APIs consist of the following APIs:

- "General Miscellaneous APIs"
- » "Service-related APIs" on page 28 «

APIs by category

APIs

These are the APIs for this 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 listed here:

- "Add Seed for Pseudorandom Number Generator (QC3ADDSD, Qc3AddPRNGSeed) API" on page 2 (QC3ADDSD, Qc3AddPRNGSeed) allows the user to add seed into the server's pseudorandom number generator system seed digest.
- "Check Communications Trace (QSCCHKCT) API" on page 3 (QSCCHKCT) returns, in bytes, the maximum size configured for the communications trace tool.
- "Control Device (QTACTLDV) API" on page 4 (QTACTLDV) provides a direct command interface to a device.
- "Convert Date and Time Format (QWCCVTDT) API" on page 8 (QWCCVTDT) allows you to convert date and time formats from one format to another format.
- "Generate Pseudorandom Numbers (QC3GENRN, Qc3GenPRNs) API" on page 16 (QC3ADDSD, Qc3GenPRNs) generates a pseudorandom binary stream.
- "Remove All Bookmarks from a Course (QEARMVBM) API" on page 18 (QEARMVBM) allows you to remove the bookmarks from a Tutorial System Support course.
- "Retrieve Data (QPARTVDA) API" on page 19 (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) API" on page 20 (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.
- "Update Device Microcode (QTAUPDDV) API" on page 24 (QTAUPDDV) provides an interface for updating device microcode from a code image stored in an Integrated File System (IFS) stream file.
- "WebFacing Environment (QqfEnvironment) API" on page 26 (QqfEnvironment) enables you to check whether a user is accessing your application through a Web browser or through 5250 emulation.

The exit program within the general miscellaneous APIs is:

- "Device Selection Exit Program" on page 41 (QIBM_QPA_DEVSEL) 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 (QC3ADDSD, 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 (OPM, QC3ADDSD; ILE, Qc3AddPRNGSeed) API allows the user to add seed into the system seed digest of the system's pseudorandom number generator.

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 system automatically generates seed using data collected from system information or by using the random number generator function on a cryptographic coprocessor, such as a 4764, 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 4764 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

Top | Cryptographic Services APIs | "Miscellaneous APIs," on page 1 | APIs by category

---

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

The caller must have *SERVICE special authority or must be authorized to the Service Trace function of the i5/OS® operating system through System i™ 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

Top | "Miscellaneous APIs," on page 1 | APIs by category

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" on page 8.

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" on page 6 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		Type	Field
Dec	Hex		
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.
'02C700'x	Device detected error: The device rejected a microcode update.
'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 Examples: Using the Control Device (QTACTLDV) API for examples of how to use the QTACTLDV API.

API introduced: V4R4

"Control Device (QTACTLDV) API" on page 4 | "Miscellaneous APIs," on page 1 | APIs by category

---

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(*)

Optional Parameter Group 1:

6	Input time zone	Input	Char(10)
7	Output time zone	Input	Char(10)
8	Time zone information	Output	Char(*)
9	Length of time zone information	Input	Bin(4)
10	Precision indicator	Input	Char(1)

Optional Parameter Group 2:

11	Input time indicator	Input	Char(1)

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 and time value to time-stamp format
- Convert a date from one character format to another
- Convert a date and time based on input and output time zone values and return the time zone information that is associated with the converted output
- Specify a precision of milliseconds or microseconds for your input and output variables
- Retrieve a current clock time based on the output time zone and return it based on the output format you specify

For additional information on converting dates and times, see "Usage Notes" on page 14.

Authorities and Locks

None.

Required Parameter Group

Input format

INPUT; CHAR(10)

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

**CURRENT*	The current system time.
**DTS*	System time-stamp.
**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)).

Input variable

INPUT; CHAR(*)

The data to be converted. If the input format is *CURRENT, then this parameter is not used. See "Input and Output Variable Formats" on page 11 to determine the structure of the input variable for all other input formats.

Output format
INPUT; CHAR(10)

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

**DTS*	System time-stamp.
**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 *DOS, the first 11 characters of this parameter are used. For details, see "DOSGetDateTime Value Structure" on page 13. See "Input and Output Variable Formats" on page 11 to determine the structure of the output variable for all other output formats.

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

Input time zone
INPUT; CHAR(10)

Specifies the time zone associated with the input variable. If the input format is *CURRENT, then this parameter is not used. The default value is *SYS. Valid values are:

**SYS*	The input variable is a local system time value and the associated time zone is specified by the time zone system value.
**UTC*	The input variable is a » Gregorian « Coordinated Universal Time (UTC) value.
**JOB*	The input variable is a local job time value and the associated time zone is specified by the time zone job attribute.
Time zone name	Specifies the name of a time zone description (*TIMZON) object.

Output time zone
INPUT; CHAR(10)

Specifies the time zone associated with the output variable. The default value is *SYS. Valid values are:

**SYS*	The output variable is a local system time value and the associated time zone is specified by the time zone system value.
**UTC*	The output variable is a » Gregorian « Coordinated Universal Time (UTC) value.
**JOB*	The output variable is a local job time value and the associated time zone is specified by the time zone job attribute.

Time zone name Specifies the name of a time zone description (*TIMZON) object.

Time zone information
OUTPUT; CHAR(*)

Specifies the time zone information associated with the output time zone. If 0 is specified for the length of time zone information, then this parameter is not used. For the format of the structure, see "Time Zone Information Value Structure" on page 13.

Length of time zone information
INPUT; BIN(4)

Specifies the length of the time zone information to be returned. The minimum length is 0 which indicates to not return any time zone information.

Precision indicator
INPUT; CHAR(1)

Specifies the precision of the input and output variables. The default value is 0 or milliseconds. Valid values are:

0 The input and output variables will have a precision in milliseconds.
1 The input and output variables will have a precision in microseconds.

Optional Parameter Group 2

Input time indicator
INPUT; CHAR(1)

Specifies which segment of time to use when the input variable has a date and time value that matches a repeated time. Otherwise, this parameter is not used. Repeated times occur when time changes from Daylight Saving Time (DST) to Standard Time (ST). For example, if DST ends on a given day at 02:00AM, then the segment of time from 01:00:00.000000 to 01:59:59.999999 on that day repeats. The first segment of time is considered in DST and the second segment is considered in ST. The default value is 1 or use the DST segment. For additional information on this parameter, see "Usage Notes" on page 14.

0 The input variable contains a date and time value that is contained in the second or Standard Time segment.
1 The input variable contains a date and time value that is contained in the first or Daylight Saving Time segment.

Input and Output Variable Formats

This table shows the format used for the input or output variable parameters.

Input or Output Format	Input or Output Variable
*DTS	System time-stamp. The first 8 characters are used.
*YYMD, *MDYY, *DMYY, *LONGJUL in milliseconds	The first 17 characters are used. See "17-Byte Character Date and Time Value Structure" on page 12.
All other character formats in milliseconds	The first 16 characters are used. See "16-Byte Character Date and Time Value Structure" on page 12.
*YYMD, *MDYY, *DMYY, *LONGJUL in microseconds	The first 20 characters are used. See "20-Byte Character Date and Time Value Structure" on page 12.
All other character formats in microseconds	The first 19 characters are used. See "19-Byte Character Date and Time Value Structure" on page 12.

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 and the precision indicator specifies milliseconds.

Offset	Description
0	Century. Possible values are 0, which indicates years 19*xx*, 1, which indicates years 20*xx* and so forth through 9, which indicates years 28*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 and the precision indicator specifies milliseconds.

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.

19-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 and the precision indicator specifies microseconds.

Offset	Description
0	Century. Possible values are 0, which indicates years 19*xx*, 1, which indicates years 20*xx* and so forth through 9, which indicates years 28*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.
13-18	Microseconds. This value cannot be blanks.

20-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 and the precision indicator specifies microseconds..

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.
14-19	Microseconds. This value cannot be blanks.

DOSGetDateTime Value Structure

This table shows the structure used for the output variable.

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 offset (in minutes)[2, 3]
10	Day of the week, where 0 is Sunday (0-6)[1]

Notes:

[1] A 1-byte integer.

[2] A 2-byte integer.

[3] This is the negative value of the offset associated with the specified output time zone. If *UTC is specified for the output time zone, then this value will be 0. If an output time zone is not specified, then this is the negative value of the system value QUTCOFFSET.

Time Zone Information Value Structure

This table shows the structure used for the time zone information output parameter. If *UTC is specified for the output time zone, or if the input and output time zone parameter values are the same and the input variable contains a date that is outside the supported date range (from August 25, 1928, 00:00:00.000000 to May 09, 2071, 00:00:00.000000), then all binary fields will be set to 0 and all character fields will be set to blanks.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Time zone description name
18	12	CHAR(1)	Reserved
19	13	CHAR(1)	Current Daylight Saving Time indicator
20	14	BINARY(4)	Current offset
24	18	CHAR(50)	Current full name
74	4A	CHAR(10)	Current abbreviated name
84	54	CHAR(7)	Current message identifier
91	5B	CHAR(10)	Message file name
101	65	CHAR(10)	Message file library
111	6F	CHAR(1)	Reserved
112	70	BINARY(4)	Year offset

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.

Current abbreviated name. The abbreviated, or short, name for the time zone. This field will contain either the Standard Time or Daylight Saving Time abbreviated name depending on whether or not Daylight Saving Time is in effect. If the time zone description uses a message to specify the current abbreviated name and the message cannot be retrieved, this field returns *N. This can occur when the caller of the API is not authorized to the message file or its library, the message file cannot be found or the message does not exist in the message file.

Current Daylight Saving Time indicator. Indicates whether or not the output date and time (output variable converted based on the output time zone) is observing Daylight Saving Time or not. Valid values that are returned are:

0 Daylight Saving Time is not being observed (Standard Time).
1 Daylight Saving Time is being observed.

Current full name. The full, or long, name for the time zone. This field will contain either the Standard Time or Daylight Saving Time full name depending on whether or not Daylight Saving Time is in effect. If the time zone description uses a message to specify the current full name and the message cannot be retrieved, this field returns *N. This can occur when the caller of the API is not authorized to the message file or its library, the message file cannot be found or the message does not exist in the message file.

Current message identifier. The identifier of the message that contains the current full and abbreviated names. This field will be *NONE if a message was not specified when the time zone description was created.

Current offset. The time difference, in minutes, between the output time zone and Coordinated Universal Time (UTC). This value has been adjusted for Daylight Saving Time, if necessary.

Message file library. The name of the library that contains the message file. The field will contain all blanks if the current message identifier is *NONE.

Message file name. The name of the file that contains the current message. The field will contain *NONE if the current message identifier is *NONE.

Reserved. An unused field.

Time zone description name. The name of the time zone description that is associated with the output time zone. If *SYS or *JOB was specified for the output time zone and a time zone has not been set for the Time zone (QTIMZON) system value, this field returns *N.

» **Year offset.** The number of years that the current year in the calendar system used with this time zone differs from the current Gregorian year. The range is -140 to 140. «

Usage Notes

When converting an input date from a 2-digit year format to a *DTS time-stamp format without time zone conversion, the supported date range is from August 23, 1928, 12:03:06.314752 (.315 for milliseconds) to May 10, 2071, 11:56:53.685240 (.685 for milliseconds). Converting an input date that is outside this range will result in an output date within this range.

When converting an input date from a 4-digit year format to a *DTS time-stamp format without time zone conversion, the supported date range is from August 24, 1928, 00:00:00.000000 to May 09, 2071, 23:59:59.999999 (.999 for milliseconds). Converting an input date that is outside this range will result in error message CPF1060.

When converting an input date from a 4-digit year format to a 2-digit year format without time zone conversion, the supported date range is from January 1, 1900, 00:00:00.000000 to December 31, 2899, 23:59:59.999999 (.999 for milliseconds). Converting an input date that is outside this range will result in error message CPF1060.

When converting an input date from a 4-digit year format to a 4-digit year format without time zone conversion, the supported date range is from January 1, 0001, 00:00:00.000000 to December 31, 9999, 23:59:59.999999 (.999 for milliseconds). Converting an input date that is outside this range will result in error message CPF1060.

When converting an input date from 2-digit year format to a 2-digit year format without time zone conversion, the supported date range is from January 1, 1900, 00:00:00.000000 to December 31, 2899, 23:59:59.999999 (.999 for milliseconds). The century digit of the input variable is copied into the output variable without validation.

When converting an input date from 2-digit year format to a 4-digit year format without time zone conversion, the supported date range is from January 1, 1900, 00:00:00.000000 to December 31, 2899, 23:59:59.999999 (.999 for milliseconds).

When converting an input date from a *DTS time-stamp format to an output date of any format without time zone conversion, the supported date range is from August 23, 1928, 12:03:06.314752 (.315 for milliseconds) to May 10, 2071, 11:56:53.685240 (.685 for milliseconds).

When converting an input date of any format to an output date of any format that involves time zone conversion as well, the supported date range is from August 25, 1928, 00:00:00.000000 to May 08, 2071, 23:59:59.999999 (.999 for milliseconds). Converting an input date that is outside this range will result in error message CPF1060.

When moving from Standard Time (ST) to Daylight Saving Time (DST) there is a window of time (1 hour) that does not occur. Any time zone conversion where the input variable date and time value is within this window will result in error message CPF1060.

When moving from Daylight Saving Time (DST) to Standard Time there is a window of time (1 hour) that repeats. For example, if DST ends on a given day at 02:00AM, then the segment of time from 01:00:00.000000 to 01:59:59.999999 on that day repeats. The first segment of repeated time is the DST segment. The second segment of repeated time is the Standard Time segment. It is possible using time zone conversion to have the output variable date and time value end up in either segment. If you are retrieving time zone information, the current Daylight Saving Time indicator will be set accordingly. By default, for any time zone conversion the input variable that is within this window of time that repeats is considered part of the DST segment. However, you can use the optional Input time indicator parameter to cause the input variable to be considered within the Standard Time segment. You can copy the resultant current DST indicator into the Input time indicator parameter when converting back and forth between time zones. For example, when converting a date and time value from *UTC to time zone A, the resultant time is 01:15:00 AM and the current DST indicator returned is 0, which means the resultant time is Standard Time. In order to obtain the original *UTC value when converting back to *UTC from time zone A, the current DST indicator value should be copied to the Input time indicator parameter. This will cause the date and time value to be treated as Standard Time rather than as the default, Daylight Saving Time.

You can convert any input format except *CURRENT to the same output format without receiving an error (time zone conversion is not specified, or if specified, the input and output time zone parameter

values must be the same and the time zone information length must be 0 as well). For these cases, the input variable is copied into the output variable without validation.

When converting one character date format (that is, anything other than *CURRENT, the current machine-clock time, *DTS, the system time-stamp, or any specified time zone conversion) to another character date format, the date information is validated and converted. However, the time portion of the input variable is copied into the output variable without validation.

When requesting time zone conversion with different input and output time zone values, or when requesting time zone information, the time portion is validated and converted as well as the date portion.

When converting a character date and time value to *DTS and back to character format using microseconds precision, there is a rounding error of minus 1 to minus 7 microseconds. If you specify a precision of microseconds, it is recommended that you use a microsecond value that is evenly divisible by 8.

» The system has the ability to manage a year offset that specifies the number of years that the current year in the calendar system used with this time zone differs from the current Gregorian year. When the input time format is *CURRENT, the date returned will be in the non-Gregorian year which takes the year offset into account. When specifying an input time zone value that is not *UTC and an output time zone value of *UTC, the year offset is removed along with the time zone offset. The converted output will be in the UTC time zone (based on the Greenwich meridian) and will be in the Gregorian year. To get a result that includes the year offset, specify an output time zone that defines Greenwich Mean Time such as the IBM-supplied Q0000GMT and that also contains the desired year offset. The converted output will be in the GMT time zone (same as UTC time zone) and will also be in the non-Gregorian year. «

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	Millisecond or microsecond value not valid.
CPF1850 E	Format &1 not valid
CPF24B4 E	Severe error while addressing parameter list.
CPF3C36 E	Number of parameters, &1, for API 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: V2R1

Top | Date and Time APIs | “Miscellaneous APIs,” on page 1 | APIs by category

---

Generate Pseudorandom Numbers (QC3GENRN, 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 (OPM, QC3GENRN; ILE, 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 system automatically generates seed using data collected from system information or by using the random number generator function on a cryptographic coprocessor, such as a 4764, 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

Top | Cryptographic Services APIs | "Miscellaneous APIs," on page 1 | APIs by category

---

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

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.

Authorities and Locks

None.

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

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" on page 21 and "PAST0200 Format" on page 21 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		Type	Field
Dec	Hex		
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		Type	Field
Dec	Hex		
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

Offset		Type	Field
Dec	Hex		
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 system 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

Top | "Miscellaneous APIs," on page 1 | APIs by category

Update Device Microcode (QTAUPDDV) API

Required Parameter Group:

1	Device name	Input	Char(10)
2	Source path name	Input	Char(*)

Optional Parameter Group:

3	Length of source path name	Input	Binary(4)
4	Format Name	Input	Char(8)
5	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: No

The Update Device Microcode (QTAUPDDV) API provides an interface for updating device microcode from a code image stored in an Integrated File System (IFS) stream file.

The QTAUPDDV API only supports self-configuring, stand alone tape devices. The Retrieve Device Capabilities (QTARDCAP) API can be used to determine if the tape device is a self-configured tape device. Other types of devices are currently not supported by 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 the tape device.

Authorities and Locks

API Public Authority
*EXCLUDE

Special Authority
*SERVICE

Directory Access Authority
*X

Stream File Access Authority
*R

Device Description Authority
*CHANGE

Device Description Lock
*EXCLRD

Required Parameter Group

Device name
INPUT; CHAR(10)

The name of the device for which for which the code image is to be updated.

Source path name.
INPUT; CHAR(*)

The path name for the code image.

Optional Parameter Group

Length of source path name.
INPUT; BINARY(4)

The length of the source path name provided. Valid values range from 1 through 32048 or -1. If this parameter is omitted, the source path name is assumed to be a simple blank terminated path name. Use -1 if the length is in the LG-type structure.

Format name
INPUT; CHAR(8)

The format of the source path name parameter. If this parameter is omitted the source path name is assumed to be a simple blank terminated path name in the current CCSID.

TAUD0100
The source path name is a simple path name in the current CCSID.

TAUD0200
The source path name is a LG-type path name structure. For more information on this structure, see Path name format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter. If this parameter is omitted, any problems will result in escape messages to the application.

Error Messages

Message ID	Error Message Text
CPFA0A9 E	Object not found. Object is *.
CPFA09C E	Not authorized to object.
CPF222E E	* special authority is required.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter * not valid.
CPF3C21 E	Format name * is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF6B35 E	File member is in use.
CPF6708 E	Command ended due to error.
CPF671F E	Parameter list not correct.
CPF67C2 E	Path name structure field * not valid.
CPF67C8 E	Command faild for device *. Reason code *.
CPF8068 E	Error detected while processing file.
CPF9802 E	Not authorized to object * in *.
CPF9814 E	Device &1 not found.

API introduced: V5R4

Top | Miscellaneous APIs | APIs by category

WebFacing Environment (QqfEnvironment) API

The WebFacing Environment (QqfEnvironment) API enables you to check whether a user is accessing your application through a Web browser or through 5250 emulation. Use this API when you would like to change the behavior of your program according to the type of access a user has. For example, there may be an extra field or different text that you would like to display if your program is being accessed through a browser, but you would like to suppress display of the field or text if 5250 emulation is being used.

The WebFacing Environment API is called QqfEnvironment and is part of the WebFacing server runtime. The external procedure name QqfEnvironment is case sensitive. It is a procedure packaged in a service program called QQFENV that is located in the QSYS library. The API returns 1 if the application is running under WebFacing and 0 if it is running under 5250 emulation.

Examples

The following examples show how to use this API. In the RPG sample, the external procedure QqfEnvironment is defined with the DSpec `QQFENV`. In this example, the `QQFENV` DSpec has been given the same name as the service program and has been defined as an integer since the procedure returns 0 or 1.

A DSpec rc has also been defined to hold the value 0 or 1 when the Eval rc = QQFENV is performed. The RPG progam then uses the value of rc to conditionally determine the behaviour of the program and what will get displayed on the DDS display.

In the DDS sample below, if the value for rc in the RPG module is NOT 1, the text "Application is not running in the Webfacing environment" will be displayed. If the value for rc is 1, the text "Application is running in the Webfacing environment" will be displayed.

When you are creating a program to use this API:

1. Use the CRTRPGMOD command to create a module with your RPG code that is calling the API. An RPG module needs to be created because it is using a procedure not in the program.
2. When you create your program (CRTPGM) use the BNDSRVPGM keyword to bind your RPG module with the QQFENV service program in QSYS.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 44.

RPGLE sample

```
.....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     FCHKENVFM  CF   E             Workstn
      *
     DQQFENV           PR            10I 0 Extproc('QqfEnvironment')
      *
     Drc               S             10I 0
      *
     C                   Eval      rc = QQFENV
     C                   Eval      FLD001 = rc
      *
     C                   Dow       NOT *IN03
      *
     C                   If        rc = 1
     C                   Eval      *in01 = *on
      *
     C                   Else
     C                   Eval      *IN01 = *off
     C                   EndIf
      *
     C                   Exfmt     FMT01
     C                   EndDo
      *
     C                   Eval      *inlr = *on
```

DDS Sample

```
....+A*..1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A*%%TS  SD  20010924  150104  USERID      REL-V4R4M0  5769-PW1
     A*%%EC
     A                                      DSPSIZ(24 80 *DS3)
     A          R FMT01
     A*%%TS  SD  20010924  150104  USERID      REL-V4R4M0  5769-PW1
     A                                      CA03(03)
     A                                   1 24'Testing Webfacing Environment API'
     A                                      DSPATR(HI)
     A                                  10  3'F3=Exit'
     A                                      COLOR(BLU)
     A N01                               5 13'Application is not running in the -
     A                                      Webfacing environment'
     A  01                               6 15'Application is running in the Webf-
     A                                      acing environment'
     A                                      COLOR(RED)
```

```
A            FLD001         4S 0O  7 39
A                                  7 22'QqfEnvironment:'
A
A*%%GP SCREEN1    01
```

API introduced: V5R1

Top | “Miscellaneous APIs,” on page 1 | APIs by category

Service-related APIs

The service-related APIs must be used only when recommended by an IBM® service representative for collecting problem-related information.

The service-related APIs are listed here:

- “Control Trace (QWTCTLTR) API” (QWTCTLTR) turns the trace function on and off.
- “Dump Device (QTADMPDV) API” on page 29 (QTADMPDV) collects information for your IBM service representative for use immediately after a suspected device and/or tape management system failure.
- “Dump Flight Recorder (QWTDMPFR) API” on page 32 (QWTDMPFR) dumps the contents of the flight recorders for jobs that have them.
- “Dump Lock Flight Recorder (QWTDMPLF) API” on page 34 (QWTDMPLF) dumps the contents of the lock flight recorder for the device that is specified in the parameter that is passed to the program.
- “Perform Miscellaneous File System Functions (QP0FPTOS) API” on page 35 (QP0FPTOS) performs a variety of file system functions.
- “Set Lock Flight Recorder (QWTSETLF) API” on page 38 (QWTSETLF) turns the lock flight recorder on and off.
- “Set Trace (QWTSETTR) API” on page 39 (QWTSETTR) starts the Trace Job (TRCJOB) command for the job passed on the job and user name parameter at the earliest point while the job is starting.

“Miscellaneous APIs,” on page 1 | APIs by category

Control Trace (QWTCTLTR) API

Required Parameter:

1	Control value	Input	Char(10)

Optional Parameter:

2	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: No

The Control Trace (QWTCTLTR) API turns the early trace function on and off. The value of *ON is passed to the program to turn the early tracing on, and *OFF is passed to turn the early tracing off.

When the early trace function is turned on, the jobs that are set up by the Set Trace (QWTSETTR) API begin tracing as soon as they are started. The tracing is stopped by turning it off with the Control Trace (QWTCTLTR) API. When *OFF or *RESET is passed to the program, this causes the trace information for the jobs to dump to spooled files.

The information set up by this API remains in effect during an initial program load (IPL).

This API should be used only when recommended by your IBM® service representative.

Authorities and Locks

None.

Required Parameter

Control value

INPUT; CHAR(10)

The value passed to turn the early trace function on or off. The valid values are:

**ON*	Turns early tracing on.
**OFF*	Turns early tracing off and stops any trace activity started by this API.
**RESET*	Turns early tracing off, stops any trace activity started by this API, and clears the space that contains the information that was set up by the QWTSETTR API.

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. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

Message ID	Error Message Text
CPF119D E	Value &1 specified for parameter 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: V2R3

Top | "Miscellaneous APIs," on page 1 | APIs by category

Dump Device (QTADMPDV) API

Required Parameter Group:

1	Device name	Input	Char(10)

Optional Parameter Group:

2	Type of information	Input	Char(10)
3	Problem identifier	Input	Char(10)
4	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Dump Device (QTADMPDV) API collects information for your IBM® service representative. This API should be used immediately after a suspected device and/or tape management system failure. If the API is not used immediately, other device operations may cause the flight recorders to wrap, which could result in lost information. A problem identifier will be created and an APAR library will be generated similar to the Save APAR Data (SAVAPARDTA) command. To save the APAR library, use Work with Problem (WRKPRB) command. Choose the option to work with the problem and then the option to save the APAR library. If an existing problem identifier is passed to this API, then the spooled files generated will be logged against that problem identifier and no new problem identifier will be generated.

The Dump Device API currently supports the following device types:

- Tape (TAP) devices
- Tape media library (TAPMLB) devices
- Optical (OPT) devices
- Optical media library (OPTMLB) devices

Note: The information provided in and the number of spooled files may change at anytime. The information provided is intended for problem determination.

The Dump Device (QTADMPDV) API dumps the following information for tape or tape media library devices, into spooled files:

- Dump of the device description for the device specified in the parameter that is passed to the program.
- Device description details of the device.
- Device capabilities (QTARDCAP output).
- QTARDINF API output.
- Licensed Internal Code tape flight recorders.
- IOP trace for the device.
- QSYSOPR message queue.
- Message queue for the user/device doing the QTADMPDV call.
- Licensed Internal Code logs from the last 24 hours.
- Product Activity Logs from the last 24 hours.
- The PTF list.
- A Work with Configuration Status (WRKCFGSTS) listing.
- Media and Storage Extensions (MSE) flight recorder.
- The history log (QHST).
- QTAPARB job log and display job (DSPJOB) information.
- Job logs and display jobs (DSPJOB) of the active jobs that have used the device as indicated in the flight recorder data.
- The job log and display job (DSPJOB) of the job that is processing this API.
- The Media library inventory information and display file description (DSPFD).
- The Media library category information and display file description (DSPFD).
- The Media library filter information and display file description (DSPFD).
- The Display Hardware Resources output (DSPHDWRSC).
- QTAHRSRV flight recorders.
- The virtual tape information area.
- The problem log summary.
- Communication information that is associated with the media library device. This includes the line, controller, and device descriptions.

Note that this API will generate multiple spooled files that may get large depending upon the job logs that are being printed and the size of the other device information. Submitting the call to batch may be used if system performance is a concern. That is, if the API is called from the system console at high priority, it may degrade performance on other critical processing. Since many and potentially large spooled files may be generated, ensure that there is enough system storage available to handle the request.

Authorities and Locks

Device description: *USE

To dump the IOP trace you need use (*USE) authority to the Control Device API (QTACTLDV).
See the documentation for this API for additional authority requirements.

To dump the PTF details you need use (*USE) authority to the Display Program Temporary Fix (DSPPTF) command.
See the documentation for this command for additional authority requirements.

To dump the VLOGS you need use (*USE) authority to the Print Internal Data (PRTINTDTA) command and Service (*SERVICE) special authority.
See the documentation for this command for additional authority requirements.

To dump the Product Activity Logs (PALS) you need use (*USE) authority to the Print Error Log (PRTERRLOG) command.
See the documentation for this command for additional authority requirements.

To dump the QTAPARB job log and display job information, you need Job Control (*JOBCTL) and All Object (*ALLOBJ) special authorities.

To dump the Media Library Inventory file description, you need use (*USE) authority to the QATAMID and QLTAMID files in QUSRSYS.

To dump the Media Library Category file description, you need use (*USE) authority to the QATACGY and QLTACGY files in QUSRSYS.

To dump the Cartridge Filter file description, you need use (*USE) authority to the QATAFTR file in QUSRSYS.

To dump the QTAHRSRV flight recorders you need use (*USE) authority to the Dump System Object (DMPSYSOBJ) command.
See the documentation for this command for additional authority requirements.

Required Parameter

Device name
INPUT; CHAR(10)

The name of the device for which debugging information is being dumped.

Optional Parameter Group

Type of information
INPUT; CHAR(10)

The type of information to be dumped.
Valid values are:

**ALL*	All information needed by IBM will be dumped to spooled files.
**MSE*	Media and Storage Extension (MSE) flight recorder will be dumped. *This is only valid for tape or tape media library devices.*

Problem identifier
INPUT; CHAR(10)

The problem identifier of the problem being analyzed. Problems with different system origins can have the same identifier. The possible values are:

**NEW*	A problem identifier will be created.
problem-identifier	The 10-character problem identifier of the problem being selected.

Error code
I/O; CHAR(*)

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

Examples

The following are examples of calls to the API from command entry:

- CALL QTADMPDV TAP01

  The dump device will dump information about TAP01 and assigns it to a created problem identifier.
- CALL QTADMPDV TAPMLB01

  The dump device will dump information about TAPMLB01 and assigns it to a created problem identifier.
- CALL QTADMPDV (TAP01 *ALL 9628851615 x'00000000')

  The dump device will dump information about TAP01 and assign it to an existing problem identifier.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF6709 E	Parameter &3 not correct.
CPF6721 E	Device &1 not a tape device.
CPF673F E	Device &1 does not support &2.
CPF9814 E	Device &1 not found.
CPF9825 E	Not authorized to device &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R1

Top | "Miscellaneous APIs," on page 1 | APIs by category

Dump Flight Recorder (QWTDMPFR) API

Optional Parameter Group:

1	Qualified job name	Input	Char(26)

Default Public Authority: *USE
Threadsafe: No

The Dump Flight Recorder (QWTDMPFR) API dumps the contents of flight recorders for jobs that have them. A **flight recorder** is an object that stores trace information to record a history of what has happened in system programs. The flight recorder contains only information that helps to identify the flow of system programs and status information. The flight recorder for a job is a temporary object and is not available after an IPL.

The following types of jobs have flight recorders:

- Subsystem monitors
- System jobs

You can use the QWTDMPFR API to collect information for your IBM® service representative. This API dumps the contents of job flight recorders to spooled files. You can then collect the files and submit them to your IBM service representative for debugging.

Authorities and Locks

Authority to use the API

To use this API, you must have service (*SERVICE) special authority

Optional Parameter Group

Qualified job name

INPUT; CHAR(26)

The name of the job whose flight recorder is to be dumped. The qualified job name has three parts:

Job name CHAR(10). A specific job name or one of the following special values:

**ACTIVE*

The flight recorders for all active system jobs and all active subsystem monitor jobs is dumped.

User name CHAR(10). A specific user profile name, or blanks when the job name is a special value.

Job number CHAR(6). A specific job number, or blanks when the job name is a special value.

Usage Notes

The QWTDMPFR API can be called with no parameters. This invocation dumps flight recorders for all active system jobs and all active subsystem monitor jobs. This is usually the best choice. You can use a **Call Program (CALL)** command from the **Command Entry** prompt.

```
CALL PGM(QSYS/QWTDMPFR)
```

The QWTDMPFR API can be called with one parameter. This allows you to dump the flight recorder for a single job. The job does not need to be active, as long as there has not been an IPL since the job was active. You can use a **Call Program (CALL)** command from the **Command Entry** prompt and use a job name parameter. The qualified job name must be specified in upper case characters because the command analyzer does not change character strings that appear between quote marks.

```
CALL PGM(QSYS/QWTDMPFR) PARM('QTAPARB   QSYS      001234')
```

Error Messages

Message ID	Error Message Text
CPF1321 E	Job &1 user &2 job number &3 not found.
CPF1332 E	End of duplicate job names.
CPF222E E	&1 special authority is required.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.

Message ID	Error Message Text
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9800 E	All CPF98xx messages could be signaled. xx is from 01 to FF.

API introduced: V2R2

Top | "Miscellaneous APIs," on page 1 | APIs by category

Dump Lock Flight Recorder (QWTDMPLF) API

Required Parameter Group:

1	Device name	Input	Char(10)

Optional Parameter Group:

2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Dump Lock Flight Recorder (QWTDMPLF) API dumps the following information into spooled files:

- The contents of the lock flight recorder for the device specified in the parameter passed to the program
- QSYSARB job log
- QLUS job log
- Job logs of the active jobs that have used the device as indicated in the lock flight recorder data
- The history log (QHST)
- Device description of the device
- Controller description of the controller to which the device is attached
- Line description of the line to which the controller is attached
- A Work with Object Locks (WRKOBJLCK) listing for the device
- A Work with Configuration Status (WRKCFGSTS) listing for the controller
- The subsystem description of active subsystems that have touched the device
- Associated internal system objects

You can use the QWTDMPLF API to collect information for your IBM® service representative.

Authorities and Locks

Authority to use the API

To use this API, you must have service (*SERVICE) special authority

Required Parameter

Device name

INPUT; CHAR(10)

The name of the device for which flight recorder information will be dumped.

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. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

Message ID	Error Message Text
CPF119C E	Value &1 specified for parameter is not valid.
CPF222E E	&1 special authority is required.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9800 E	All CPF98xx messages could be signaled. xx is from 01 to FF.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Top | "Miscellaneous APIs," on page 1 | APIs by category

Perform Miscellaneous File System Functions (QP0FPTOS) API

Required Parameter Group:

1	Function type	Input	Char(*)
2	Function extension 1	Input	Char(*)
3	Function extension 2	Input	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Perform Miscellaneous File System Function (QP0FPTOS) API is used to perform a variety of file system functions. The first parameter defines the type of function that is requested. Other parameters are optional, depending on the selected function. The output from this API varies, based on the selected function. See the function descriptions for more details.

Authorities and Locks

To call this program you must have *SERVICE special authority, or be authorized to the Service Dump function of i5/OS® through System i™ Navigator's Application Administration support. The Change Function Usage (CHGFCNUSG) command or Change Function Usage Information (QSYCHFUI) API, with a function ID of QIBM_SERVICE_DUMP, also can be used to change the list of users allowed to perform dump operations.

Note: Adopted authority is not used.

Required Parameter Group

Required parameters vary according to the selected function. The selected function is identified by the first parameter on the call to the API.

Function Type

INPUT; CHAR(*)

The desired file system function to perform. Valid values follow:

*(1) *DUMP*

Creates a general file system dump in a spooled file with file name ″QSYSPRT″ and with ″QP0FDUMP″ in the User Data field. No other parameters are required or supported when *DUMP is specified.

*(2) *DUMPALL*

Creates a variety of file system dumps in a single spooled file with file name ″QSYSPRT″ and with ″QP0FDUMP″ in the User Data field. The following table describes the optional parameter when *DUMPALL is specified.

Function	Function extension 1	Function extension 2	Description
*DUMPALL	Job number (CHAR 6)	(Not supported)	Specifies the job that is dumped. If a job is not specified, the data is dumped for all jobs. If there are multiple jobs with the same number, the first one encountered will be dumped.

*(3) *DUMPLFS*

Creates a dump of logical file system data in a spooled file with file name ″QSYSPRT″ and with ″QP0FDUMP″ in the User Data field. The following table describes the optional parameter when *DUMPLFS is specified.

Function	Function extension 1	Function extension 2	Description
*DUMPLFS	Job number (CHAR 6)	(Not supported)	Specifies the job that is dumped. If a job is not specified, the data is dumped for all jobs. If there are multiple jobs with the same number, the first one encountered will be dumped.

*(4) *NFSFORCE*

Sets various values and modes for the network file system. The following table describes the required parameters when *NFSFORCE is specified.

Function	Function extension 1	Function extension 2	Description
*NFSFORCE	V2	ON or OFF	If ON, indicates version 2 mounts only by the client. If QNFSMNTD is started afterwards, then server will permit version 2 mounts only.

*(5) *REBUILDDEVNULL*

Attempts to create the /dev/null and dev/zero character special files. If an existing dev/null or dev/zero object exists that is not a character special file, then the object is renamed to /dev/null.prv or dev/zero.prv. If /dev/null.prv or /dev/zero.prv exists, then it it renamed to /dev/null.prv.001 or /dev/zero.prv.001, /dev/null.prv.002 or /dev/zero.prv.002, and so on, until a name is found for the object. If 999 is exceeded and the rename cannot be done, the object is not renamed and an informational message is issued and the QP0FPTOS program completes successfully. No other parameters are required or supported when *REBUILDDEVNULL is specified.

*(6) *TRACE6ON or *TRACE6OFF*

*TRACE6ON starts the logging of trace messages in the user job log for some network file system functions. *TRACE6OFF stops the logging of these messages.

*(7) *TRACE8ON or *TRACE8OFF*

*TRACE8ON starts the logging of trace messages to the QSYSOPR message queue for some network file system functions. *TRACE8OFF stops the logging of these messages.

*(8) *TRACE9ON or *TRACE9OFF*

*TRACE9ON starts the collection of some network file system statistics and resets the statistics. *TRACE9OFF stops the collection of these statistics.

*(9) *DUMPNFSSTATS*

Creates a file system dump of network file system (NFS) statistics (both client and server) in a spooled file with file name ″QSYSPRT″ and with ″QP0FDUMP″ in the User Data field. The information dumped comes from a window of time specified with the *TRACE9ON/OFF function. No other parameters are required or supported when *DUMPNFSSTATS is specified.

Function extension 1

INPUT; CHAR(*)

Function extension 1 is optional or required, based on the first parameter. Whenever it is valid, function extension 1 is described above along with a first parameter description. Function extension 1 is valid when the first parameter is listed below:

*(1) *DUMPALL*

*(2) *DUMPLFS*

*(3) *NFSFORCE*

Function extension 2

INPUT; CHAR(*)

Function extension 2 is optional or required, based on the first parameter. Whenever it is valid, function extension 2 is described above along with a first parameter description. Function extension 2 is valid when the first parameter is listed below:

*(1) *NFSFORCE*

Usage Notes

If this API is called without the first parameter that is required, then message CPFBC53 is issued to the caller. This message specifies a parameter that is not valid. To recover, the caller is pointed to the API documentation.

Error Messages

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.

Message ID	Error Message Text
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFA0A0 E	Object name already exists.
CPFA0D4 E	File system error occurred. Error number &1.
CPDA0FF E	Program not called. You need *SERVICE authority to call this program.
CPFBC53 E	Invalid parameter.
CPFBC54 E	Not authorized to call program.

Examples

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 44.

```
CALL QP0FPTOS *DUMP
CALL QP0FPTOS (*DUMPALL '055229')
CALL QP0FPTOS (*DUMPLFS '055229')
CALL QP0FPTOS (*NFSFORCE V2 ON)
CALL QP0FPTOS *REBUILDDEVNULL
CALL QP0FPTOS *TRACE6ON
CALL QP0FPTOS *TRACE6OFF
CALL QP0FPTOS *TRACE8ON
CALL QP0FPTOS *TRACE8OFF
CALL QP0FPTOS *TRACE9ON
CALL QP0FPTOS *TRACE9OFF
CALL QP0FPTOS *DUMPNFSSTATS
```

API introduced: V5R2

Top | "Miscellaneous APIs," on page 1 | APIs by category

Set Lock Flight Recorder (QWTSETLF) API

Required Parameter:

1	Set value	Input	Char(4)

Optional Parameter:

2	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: No

The Set Lock Flight Recorder (QWTSETLF) API turns the lock flight recorder on and off. The value of *ON is passed to the program to turn the lock flight recorder on, and *OFF is passed to turn the lock flight recorder off.

When the lock flight recorder is turned on, the system will begin logging successful lock operations on devices in the lock flight recorder for the device being locked.

This API should be used only when recommended by your IBM® service representative.

Authorities and Locks

None.

Required Parameter

Set value
INPUT; CHAR(4)

The value passed to turn the lock flight recorder on or off. The valid values are:

**ON*	Turn flight recorder on.
**OFF*	Turn flight recorder off.

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. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

Message ID	Error Message Text
CPF119B E	Value &1 specified for parameter is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9800 E	All CPF98xx messages could be signaled. xx is from 01 to FF.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Top | "Miscellaneous APIs," on page 1 | APIs by category

Set Trace (QWTSETTR) API

Required Parameter Group:

1	Job name	Input	Char(10)
2	User name	Input	Char(10)

Optional Parameter:

3	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Set Trace (QWTSETTR) API starts a Trace Job (TRCJOB) command for the job passed on the parameter at the earliest point while the job is starting. This allows tracing of jobs early in the life of a job to help debug problems that could not have been done earlier because a user could not enter the command until the job was actually started.

The QWTSETTR API sets up the information about the job so that when that job is started a trace will begin.

The QWTSETTR API can be called multiple times to set up traces for multiple jobs. When the tracing is finished, the Control Trace (QWTCTLTR) API should be called using the *RESET value for the control value parameter to clear out all the job names. The Control Trace (QWTCTLTR) API must be called to turn on this trace activity.

The information set up by the QWTSETTR API will be in effect during an initial program load (IPL).

The information set up by the QWTSETTR API does not work for active jobs, but only for jobs that have not started yet.

If a job ends while the trace activity is running for that job, the trace information will be dumped to a spooled file.

The Trace Job (TRCJOB) command is issued in the job as if a user had typed in the command; therefore, the user (user name parameter) must have authorization to the TRCJOB command for this to work properly.

This API should be used only when recommended by an IBM® service representative for collecting information for problems that occur early in job initiation.

Authorities and Locks

The user (user name parameter) must have authorization to the TRCJOB command for this to work properly.

Required Parameter Group

Job Name
INPUT; CHAR(10)

The name of the job that will be traced.

User Name
INPUT; CHAR(10)

The name of the user that will be traced.

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. If this parameter is omitted, diagnostic and escape messages are issued to the application.

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.

API introduced: V2R3

Top | "Miscellaneous APIs," on page 1 | APIs by category

Exit Programs

These are the Exit Programs for this category.

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.
   - Specify a second controller naming pattern to be used to 'count' the number of existing virtual devices.
   - Specify the number of devices that can exist on the virtual controllers whose naming pattern is specified.

Authorities and Locks

You must have *ALLOBJ authority to register an exit program for the QIBM_QPA_DEVSEL exit point.

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 Format" on page 42.

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 Allow connection request.
1 Do not 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

Top | "Miscellaneous APIs," on page 1 | APIs by category

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

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

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

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

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

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

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

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

Programming interface information

This API descriptions publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced 36
Advanced Function Presentation
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
AS/400
BCOCA
C/400
COBOL/400
Common User Access
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
Enterprise Storage Server
eServer
FlashCopy
GDDM
i5/OS
IBM
IBM (logo)
InfoColor
Infoprint
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
Lotus
Lotus Notes
MO:DCA
MVS
Net.Data
NetServer
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
POWER5+
PowerPC
Print Services Facility
PrintManager
PROFS
RISC System/6000
RPG/400
RS/6000

SAA
SecureWay
SOM
System i
System i5
System Object Model
System/36
System/38
System/390
TotalStorage
VisualAge
WebSphere
xSeries
z/OS

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

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

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

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

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

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

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

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER

EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA