System i

Programming
Virtual Terminal APIs

Version 6 Release 1

System i

Programming
Virtual Terminal APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in “Notices,” on page 25.

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

Virtual Terminal APIs

The virtual terminal APIs enable application programs to interact with a System i™ platform that is performing workstation input/output (I/O).

A *virtual terminal* is a device that does not have hardware associated with it. It forms a connection between your application and i5/OS® applications, representing a physical workstation (possibly on a remote system). The i5/OS operating system manages the virtual terminal, which directs workstation I/O performed by an i5/OS application to the virtual terminal. The virtual terminal APIs allow another i5/OS application, called a *server program*, to work with the data associated with the virtual terminal.

In a distributed systems environment, the requesting program is called a *client*; the answering program is called a *server*. The client and server programs may reside on the same system or may be distributed between two different systems. The server program generally runs on behalf of (or in conjunction with) the client program. Together, the server program and the client program allow a workstation to be supported as if the workstation were connected locally.

You must specify a workstation type when a virtual terminal device is opened. A server program can select several different types of workstations. See "Open Virtual Terminal Path (QTVOPNVT) API" on page 2 for a list of the types of workstations that are supported.

Select the following topics for more information:

The virtual terminal APIs are:

- "Close Virtual Terminal Path (QTVCLOVT) API" on page 2 (QTVCLOVT) closes one or all open virtual terminal paths.
- "Open Virtual Terminal Path (QTVOPNVT) API" on page 2 (QTVOPNVT) opens a path to a virtual terminal.
- "Read from Virtual Terminal (QTVRDVT) API" on page 11 (QTVRDVT) reads data from the virtual terminal into a data buffer.
- "Send Request for i5/OS Function (QTVSNDRQ) API" on page 14 (QTVSNDRQ) sends a request to perform a particular function.
- "Write to Virtual Terminal (QTVWRTVT) API" on page 15 (QTVWRTVT) writes data to a virtual terminal from a data buffer.

Top | APIs by category

APIs

These are the APIs for this category.

Close Virtual Terminal Path (QTVCLOVT) API

Required Parameter Group:

1	Virtual terminal handle	Input	Char(16)
2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Close Virtual Terminal Path (QTVCLOVT) API closes one or all open virtual terminal paths. To close all open virtual terminal paths, the handle must be set to zero.

Authorities and Locks

None.

Required Parameter Group

Virtual terminal handle
INPUT; CHAR(16)

The reference code for the open virtual terminal path, created with the Open Virtual Terminal Path (QTVOPNVT) API. If this parameter is set to zero, all open virtual terminal paths are closed.

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
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF87F2 E	Virtual terminal handle &1 not valid.
CPF87F7 E	Parameter value &1 not valid.
CPF87F8 E	Unexpected internal system error occurred in program &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Open Virtual Terminal Path (QTVOPNVT) API

Required Parameter Group:

1	Virtual terminal handle	Output	Char(16)
2	Keyboard language type	Input	Char(3)
3	Character set	Input	Binary(4)
4	Code page	Input	Binary(4)
5	Workstation type	Input	Binary(4)
6	Qualified data queue name	Input	Char(20)
7	Key value	Input	Char(*)
8	Length of key value	Input	Binary(4)
9	Error code	I/O	Char(*)

Optional Parameter Group: 1

10	Open operation information	Input	Char(10)

Optional Parameter Group: 2

11	Session initiation information	Input	Char(*)

Optional Parameter Group: 3

12	Open feedback information	I/O	Char(*)
13	Length of open feedback information	Input	Binary(4)

Default Public Authority: *USE
Threadsafe: No

The Open Virtual Terminal Path (QTVOPNVT) API opens a path to a virtual terminal, allowing a server program to interact with an i5/OS® application. The virtual terminal path remains open until it is explicitly closed or the job is ended.

When you call the QTVOPNVT API, the operating system selects or automatically configures a virtual terminal for you and indicates that the device is logically turned on. The operating system sends a message to the specified data queue to signal the server program that data is available.

Authorities and Locks

Library Authority
*USE

User Queue Authority
*CHANGE

User Queue Lock
*EXCLRD

Required Parameter Group

Virtual terminal handle
OUTPUT; CHAR(16)

A reference code created by the operating system to identify this open virtual terminal path in later calls to other virtual terminal APIs.

Keyboard language type
INPUT; CHAR(3)

The keyboard language type for the virtual terminal. To use the system value, specify blanks for this parameter. For a list of other valid values, see the Create Device Description (Display) (CRTDEVDSP) command. For details about supported languages, see the i5/OS globalization topic collection.

Character set
INPUT; BINARY(4)

The graphic character set for the virtual terminal. Valid values are a specific graphic character set number and these special values:

0 The graphic character set system value is used.
-1 The keyboard language type is used to select the appropriate character set.

For details about the graphic character sets you can specify, see i5/OS globalization.

Note: The graphic character set system value is obtained from the default graphic character set and code page (QCHRID) system value.

Code page
INPUT; BINARY(4)

The code page for the virtual terminal. For details about the code pages you can specify, see i5/OS globalization. If you specified 0 or -1 for the character set parameter, you do not have to specify this parameter. When you use 0 for the character set parameter, the system value is used for this parameter. When you use -1 for the character set parameter, the code page value is derived from the keyboard language type parameter.

Note: The code page system value is obtained from the default graphic character set and code page (QCHRID) system value.

Workstation type
INPUT; BINARY(4)

The type of workstation to use. Valid values are 1 through 15. See "Supported Workstation Types and Models" on page 10 for an explanation of the values.

Other workstation types and models are supported. You can specify these by determining their equivalents in Workstation Types and Models (page 10).

If a virtual terminal description does not yet exist for the workstation type specified, the operating system attempts to configure the workstation automatically. Automatic configuration is controlled by the Automatic virtual device configuration (QAUTOVRT) system value, which specifies the number of virtual terminals that the operating system can configure automatically. See for more information about the QAUTOVRT value.

Qualified data queue name
INPUT; CHAR(20)

The name and library of the data queue used by your application program to receive data from the operating system asynchronously. The first 10 bytes are for the data queue name; the second 10 bytes are for the library name. Allowable special values are:

**CURLIB*	The jobs current library
**LIBL*	The library list

Key value
INPUT; CHAR(*)

The key value to use for the data queue.

Length of key value
INPUT; BINARY(4)

The length of the key value. Valid values are 0 through 256. If you specify 0, no key is used for data queue entries.

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

Open operation information
INPUT; CHAR(10)

Information about the open operation. The characters and their meanings are:

1 Whether the PC text-assist function is supported. Valid values are:

Blank The PC text-assist function is supported.

0 The PC text-assist function is supported.

1 The PC text-assist function is not supported. The adapted word processing function is used automatically.

2-10 Reserved. These characters must be blank.

Optional Parameter Group 2

Session initiation information
INPUT; CHAR(*)

Information about the initiation of the virtual terminal session. The information must be in the following format:

Number of variable length records
The total number of all of the variable length records.

Variable length records
The attributes of the session initiation that are to be performed or changed. For the specific format of the variable length record, see "Format for Variable Length Record."

Optional Parameter Group 3

Open feedback information
I/O; CHAR(*)

A pointer to information about the device assigned to this virtual terminal session or the reason code when an automatic sign-on request fails. The layout of this information is described in "Format for Open Feedback Information" on page 9.

Length of open feedback information
INPUT; BINARY(4)

The size of the open feedback output area being supplied by the caller. The Open Feedback Information returned will be restricted such that it always returns a total number of bytes equal to or less than this parameter.

Format for Variable Length Record

The following table defines the format for the variable length records.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Key
4	4	BINARY(4)	Length of data
8	8	CHAR(*)	Data

If the length of the data is longer than the key fields data length, the data will be truncated at the right. No message will be issued.

If the length of the data is smaller than the key field s data length, the data will be padded with blanks at the right. No message will be issued.

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

Field Descriptions

Data. The data used to control the session initiation.

Key. An attribute of the session initiation. See Keys for the list of valid keys.

Length of data. The length of the data used to control the initiation.

Keys

The following table shows the valid keys for the key field area of the variable length record.

Key	Type	Field
1	CHAR(10)	User profile
2	CHAR(10)	User password
3	CHAR(10)	Initial program
4	CHAR(10)	Initial menu
5	CHAR(10)	Current library
6	CHAR(10)	Virtual device name
7	CHAR(20)	» IPv4 Network address «
8	CHAR(*)	Automatic sign-on (see “Automatic Sign-on Key” on page 8)
9	CHAR(32)	Profile Token
» 10	CHAR(28)	IPv6 Network address «

Field Descriptions

Automatic sign-on. The values needed to have a user automatically signed on once a terminal session has been established. If key 8 is not specified, automatic sign-on will not occur for this virtual terminal session; keys 3, 4, and 5 will be ignored. The automatic sign-on must be used in place of key 1 and 2 whenever the system value QPWDLVL is set to 2 or 3. When the system value QPWDLVL is set to 0 or 1, either keys 1 and 2 or key 8 may be used for automatic sign-on.

Current library. The current library to be used when using automatic sign-on. The special value *USRPRF can be used to indicate that the current library found in the user profile specified with key 1 should be used. If key 5 is not specified, *USRPRF is the default. This value is supported for both key 1 and key 8.

Initial menu. The initial menu to be used when using automatic sign-on. The special value *USRPRF can be used to indicate that the initial menu found in the user profile specified with key 1 should be used. This value is supported for both key 1 and key 8. If key 4 is not specified, *USRPRF is the default.

Initial program. The initial program to be used when using automatic sign-on. The special value *USRPRF can be used to indicate that the initial program found in the user profile specified with key 1 should be used. If key 3 is not specified, *USRPRF is the default. This value is supported for both key 1 and key 8.

IPv4 Network address. An address that is uniquely assigned to each terminal session and is used in all communications with the session. This address is associated with the virtual terminal device by the applicaion and can be accessed by any other application using the Retrieve Device Description (QDCRDEVD) API. The following format defines the layout of the network address:

1. Size of network address

CHAR(1) The number of bytes of network address actually used. All 20 bytes allocated for the network address may not actually be used.

2. Family or protocol

CHAR(1) The family or protocol that is being used (only IPv4 is supported). The family or protocol defines the layout used for the remainder of the network address.

Internet Protocol version 4 value is X'02.'

3. Internet Protocol (IP)

CHAR(2) TCP port number
CHAR(4) internet address

» IPv6 Network address. An address that is uniquely assigned to each terminal session and is used in all communications with the session. This address is associated with the virtual terminal device by the applicaion and can be accessed by any other application using the Retrieve Device Description (QDCRDEVD) API. The following format defines the layout of the network address:

1. Size of network address

CHAR(1) The number of bytes of network address actually used. All 28 bytes allocated for the network address may not actually be used.

2. Family or protocol

CHAR(1) The family or protocol that is being used (only IPv6 is supported). The family or protocol defines the layout used for the remainder of the network address.

Internet Protocol version 6 value is X'18.'

3. Internet Protocol (IP)

CHAR(2) TCP port number
CHAR(4) Flow Info - optional: Must be set to X'00000000' if not used.
CHAR(16) internet address up to 16 hex chars of address, right adjusted, unused leading chars must be set to x'00'
CHAR(4) Scope ID - optional: Must be set to X'00000000' if not used. «

Profile Token. A 32-byte Profile Token which may be used as an alternative to using key 8, or keys 1 and 2 for Automatic sign-on purposes. When key 9 is used for Automatic sign-on, keys 3, 4 and 5 will also be honored if specified. Also, if Optional parameter group 3 is specified, the Automatic sign-on response code can be used to determine the result of the Automatic sign-on request when using key 9. Profile tokens may be used for all values of the system value QPWDLVL.

User password. The user password specified for initiating this virtual terminal session in conjunction with the specified profile. If the user profile is *CURRENT, you can use the special value *NONE for the user password. If you specify a user profile other than *CURRENT or blank and do not specify a user password, an error message is returned. This value works in association with key 1 (but not with key 8). This key is supported only when system value QPWDLVL is set to 0 or 1. Note that the Automatic sign-on Reason Code value from Optional Parameter Group 3 is undefined when using keys 1 and 2 for Automatic sign-on.

User profile. The user profile specified for initiating this virtual terminal session. You can use the special value *CURRENT for the user profile. If the user profile is *CURRENT, you can use the special value

*NONE for the password. This value is used in association with key 2 (but not with key 8). If key 1 is not specified, automatic sign-on will not occur for this virtual terminal session; keys 2, 3, 4, and 5 will be ignored. This key is supported only when system value QPWDLVL is set to 0 or 1. Note that the Automatitic sign-on Reason Code value from Optional Parameter Group 3 is undefined when using keys 1 and 2 for Automatic sign-on.

Virtual device name. The specific virtual device to be associated with the terminal session.

The device is created by the system, if it does not exist, when the QAUTOVRT system value allows for this.

If no value is supplied by the caller, the virtual terminal API defaults to using the virtual device selection methods.

The device description name must be a valid *VRT and must adhere to standard i5/OS naming conventions.

Automatic Sign-on Key

The following table defines the format for the automatic sign-on key.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Sign-on length
4	4	BINARY(4)	Passphrase CCSID
8	8	BINARY(4)	Passphrase offset
12	C	BINARY(4)	Passphrase length
16	F	CHAR(8)	Client seed
24	18	CHAR(8)	Server seed
32	20	CHAR(10)	Bypass user profile
42	2A	CHAR(2)	Reserved
44	2C	CHAR(*)	Passphrase

Field Descriptions

Bypass user profile. The user profile to be automatically signed on. This profile is not interchangeable with the user profile from key 1. If key 8 is used, then the profile must be part of the automatic sign-on information.

Client seed. A randomly-generated seed that was used to encrypt the password or passphrase using the encryption level (either DES-7 or SHA-1) on the system. If a zero-length seed is provided, it is assumed that a clear-text unencrypted password or passphrase is being provided.

Passphrase CCSID. The CCSID of the password or passphrase being provided for automatic sign-on.

Passphrase offset. The offset in this structure of the actual password or passphrase being provided for automatic sign-on.

Passphrase length. The number of bytes in the password or passphrase being provided.

Passphrase. The password or passphrase associated with the user profile to be automatically signed on. This field can range from 1 to 128 bytes. If your system is pre-V5R1, or has system value QPWDLVL set to 0 or 1, then the maximum size of this field is 10 bytes.

Reserved. Reserved for future use.

Server seed. A randomly-generated seed that was used to encrypt the password or passphrase using the encryption level (either DES-7 or SHA-1) on the system. If a zero-length seed is provided, it is assumed that a clear-text unencrypted password or passphrase is being provided.

Sign-on length. The total number of bytes of the automatic sign-on data provided.

Format for Open Feedback Information

The following table defines the format for the open feedback information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of bytes available
4	4	BINARY(4)	Number of bytes returned
8	8	BINARY(4)	Reason code (see "Automatic Sign-on Reason Codes")
12	C	CHAR(10)	Device name
22	16	CHAR(4)	Reserved

Automatic Sign-on Reason Codes

The following table shows the reason codes that may be returned for automatic sign-on failure. The reason code field is undefined when using keys 1 and 2 for Automatic sign-on.

Reason Code	Failure Description
1	System error (unknown error)
2	User profile unknown
3	User profile disabled
4	Password or passphrase not valid
5	Password or passphrase is expired
6	Pre-V2R2 password
8	Next password or passphrase that is not valid will revoke user profile
9	Profile Token invalid or expired

Field Descriptions

Device Name. The device name that has actually been assigned to this terminal session.

Number of bytes available. The total number of bytes of feedback information available for return. This space is provided by the caller.

Number of bytes returned. The total number of bytes of feedback information actually returned to the caller.

Reason code. Reason an automatic sign-on request has failed. For possible reason code values, see "Automatic Sign-on Reason Codes" on page 9.

Reserved. Reserved for future use.

Supported Workstation Types and Models

The following table details the values you can specify for the QTVOPNVT APIs workstation type parameter.

Workstation Types and Models			
Value	**Workstation Type and Model**	**Equivalent Type and Model**	**Description**
1	5251-11		24 x 80 monochrome display.
2	5291-1	5291-2	24 x 80 monochrome display.
3	5292-2		24 x 80 color graphics display. This type is also emulated by a graphics workstation feature.
4	5555-B01	5555-E01	» 24 x 80 or 27 x 132 monochrome double-byte character set (DBCS) display. This type is emulated by a monochrome workstation feature that supports a DBCS display. «
5	3196-A1	3196-A2 3196-B1 3196-B2 3476-EA	24 x 80 monochrome display. This type is emulated by a monochrome workstation feature. This is what the ASCII devices emulate.
6	3179-2	3197-C1 3197-C2 3476-EC 5292-1	24 x 80 color display. This type is emulated by a color workstation feature.
7	3180-2	3197-D1 3197-D2 3197-W1 3197-W2	27 x 132 monochrome display.
8	3477-FC		27 x 132 wide-screen color display.
9	3477-FG	3477-FA 3477-FD 3477-FE 3477-FW	27 x 132 wide-screen monochrome display.
10	5555-C01	5555-F01	» 24 x 80 or 27 x 132 color double-byte character set (DBCS) display. This type is emulated by a color workstation feature that supports a DBCS display. «
11	5555-G01		» 24 x 80 or 27 x 132 double-byte character set (DBCS) monochrome graphics display. This type is emulated by a monochrome workstation feature that supports a DBCS display. «
12	5555-G02		» 24 x 80 or 27 x 132 color double-byte character set (DBCS) graphics display. This type is emulated by a color graphics workstation feature that supports a DBCS display. «
13	3486-BA		24 x 80 monochrome display.
14	3487-HA	3487-HG 3487-HW	24 x 80 or 27 x 132 monochrome display.
15	3487-HC		24 x 80 or 27 x 132 color display.

Usage Notes

1. All 5250 workstations, except 5555 Model B01, can operate as 5251 Model 11 workstations.
2. Selecting double-byte character set (DBCS) workstations requires the i5/OS primary language to be one of the DBCS national language versions (NLVs).

Error Messages

Message ID	Error Message Text
CPF1842 E	Cannot access system value &1.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C81 E	Value for key &1 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C84 E	Key &1 required with value specified for key &2.
CPF3C86 E	Required key &1 not specified.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF87D7 E	Cannot automatically select virtual device.
CPF87FA E	Character identifier not valid.
CPF87FB E	Cannot exceed maximum number of active Virtual Terminal sessions.
CPF87F0 E	Virtual terminal type value &1 not valid.
CPF87F1 E	Queue key length &1 not valid.
CPF87F2 E	Virtual terminal handle &1 not valid.
CPF87F7 E	Parameter value &1 not valid.
CPF87F8 E	Unexpected internal system error occurred in program &1.
CPF87F9 E	Keyboard language type &1 not valid.
CPF9E18 E	Attempt made to exceed usage limit for product &1. User not added.
CPF9E71 E	Grace period expired. Requesting user not added.
CPF9E73 E	Expiration date &4 was reached.
CPF9E78 E	The license key for product &1, license term &2, feature &3 is no longer valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Read from Virtual Terminal (QTVRDVT) API

Required Parameter Group:

1	Virtual terminal handle	Input	Char(16)
2	Read information	Output	Char(10)
3	Data buffer	Output	Char(*)
4	Number of bytes to read	Input	Binary(4)
5	Data received	Output	Binary(4)
6	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Read from Virtual Terminal (QTVRDVT) API reads data from the virtual terminal into the server program's data buffer. Your application should read data only if it has received an asynchronous notification message on the data queue, or if the more data flag was set on a previous read operation. The data received is in 5250 data stream format.

Only one full-screen display of data can be received at a time. If the data buffer is too small, partial displays are received and the more data flag for the QTVRDVT API's read information parameter is set to 1.

Before working with 5250 data streams, be sure to see the *IBM® 5494 Functions Reference* manual, SC30-3533. This manual can be viewed online through the IBM Publications Center  .

Authorities and Locks

None.

Required Parameter Group

Virtual terminal handle
INPUT; CHAR(16)

The reference code for the open virtual terminal path, created by the operating system with the Open Virtual Terminal Path (QTVOPNVT) API.

Read information
OUTPUT; CHAR(10)

Information about the read operation. The characters and their meanings are:

1 The operation code, which gives the server program additional information about i5/OS® status and what is expected of the server program. Valid values for this parameter are:

- *1* Invite
- *2* Output only
- *3* Put/get
- *4* Save display
- *5* Restore display
- *6* Read immediate
- *8* Read display
- *A* Cancel invite
- *B* Turn on message light
- *C* Turn off message light

For detailed descriptions of these codes, see "Read Operation Codes" on page 13.

2 More data flag. Valid values for this parameter are:

- *0* There is no more data.
- *1* More data is available. Issue the read operation again to receive the additional data. This flag is set if the buffer specified on input is not large enough to hold all of the data received from the virtual terminal.

3 Key flag. Valid character values for this parameter are:

- *0* The Enter key was pressed.
- *1* The System Request key was pressed.

4-10 Reserved. These characters must be blank.

Data buffer
OUTPUT; CHAR(*)

The server program's buffer for receiving data from the virtual terminal. The data is a 5250 data stream.

The QTVRDVT API does not lock the data buffer. Thus, other applications should not use the buffer while the API is using it.

The data buffer should be large enough to hold the largest display of data expected. If it is not large enough for all the data to fit, the more data flag of the read information parameter is set to 1. Additional read requests must be performed, until all the remaining data is received and the more data flag is set back to 0.

Number of bytes to read
INPUT; BINARY(4)

The number of bytes to read from the data buffer. This number must be smaller than or equal to the size of the data buffer.

Data received
OUTPUT; BINARY(4)

The amount of data received from the virtual terminal in bytes. If no data is received from the virtual terminal, 0 is returned. Some read operations do not return any data.

For graphic work stations, a maximum of 24 576 (24KB) bytes of data can be returned.

Error code
I/O; CHAR(*)

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

Read Operation Codes

The following table describes the operation codes that can be returned on a read request.

Read Operation Codes

Value	Response Name	Description
1	Invite	The operating system and the application are ready to receive data. The server program is expected to follow this with a write operation when data becomes available from the client program.
2	Output only	This read request returned some data. The server program should send the data to the client program. However, the operating system is not ready to receive data from the server program. The server program should not request any data from the client program yet. This response usually occurs because an application is performing several put operations to the virtual terminal device. After the last put operation by the application, a put/get operation code is usually returned on the read operation.
3	Put/get	Data is returned from this read request and should be sent by the server program to the client program. The operating system is ready to receive data from the server program. The server program should wait for data from the client program.
4	Save display	The operating system expects the server program to obtain the data from the current display and write the data to the virtual terminal. The operating system saves the display for later use, such as returning the display to the server program. The server program must indicate a save-display response on the write operation and send the saved display as data (that is, the saved display must be in the data buffer).
5	Restore display	The data returned is a previously saved display. The server program should send the data to the client program.
6	Read immediate	No data is returned from this read request. The operating system expects the server program to write data to the virtual terminal. Only data from input fields should be written.

Read Operation Codes		
Value	**Response Name**	**Description**
8	Read display	No data is returned from this read request. The operating system expects the server program to write data to the virtual terminal. The current display should be written.
A	Cancel invite	No data is returned from this read request. The operating system expects the server program to signal the client program to cancel the outstanding invite operation. When it is canceled, the server program must perform a write operation to the virtual terminal and indicate a cancel invite response. Because the response has no data associated with it, the number of bytes to write must be set to 0 for the write operation.
B	Turn on message light	No data is returned from this read request. A message has been received, and the user should be notified. The server program should signal the client program to turn on a display message indicator light or some other indicator.
C	Turn off message light	No data is returned from this read request. The display message light or some other indicator should be set off.

Error Messages

Message ID	**Error Message Text**
CPF3C90 E	Literal value cannot be changed.
CPF87F2 E	Virtual terminal handle &1 not valid.
CPF87F3 E	Data buffer length &1 not valid.
CPF87F7 E	Parameter value &1 not valid.
CPF87F8 E	Unexpected internal system error occurred in program &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Send Request for i5/OS Function (QTVSNDRQ) API

Required Parameter Group:

1	Virtual terminal handle	Input	Char(16)
2	Request	Input	Binary(4)
3	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Send Request for i5/OS® Function (QTVSNDRQ) API sends a request to the operating system to perform a particular function.

Authorities and Locks

None.

Required Parameter Group

Virtual terminal handle
INPUT; CHAR(16)

The reference code for the open virtual terminal path, created with the Open Virtual Terminal Path (QTVOPNVT) API.

Request

INPUT; BINARY(4)

The request to be processed by the operating system. Valid binary values are:

1 Cancel Previous Request: Allows the server program to cancel the previous i5/OS request. This is similar to selecting option 2 on the System Request menu.

2 Send Break Message: Causes the operating system to issue a break message to the virtual terminal. You can use this to determine whether the virtual terminal is still active.

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
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF87F2 E	Virtual terminal handle &1 not valid.
CPF87F6 E	Request value &1 not valid.
CPF87F7 E	Parameter value &1 not valid.
CPF87F8 E	Unexpected internal system error occurred in program &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Write to Virtual Terminal (QTVWRTVT) API

Required Parameter Group:

1	Virtual terminal handle	Input	Char(16)
2	Write information	Input	Char(10)
3	Data buffer	Input	Input
4	Number of bytes to write	Input	Binary(4)
5	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Write to Virtual Terminal (QTVWRTVT) API writes data from a server program's data buffer to a virtual terminal. You can send one display to the virtual terminal during each write operation. You cannot send partial or multiple displays.

Authorities and Locks

None.

Required Parameter Group

Virtual terminal handle
INPUT; CHAR(16)

The reference code for the open virtual terminal path, created with the Open Virtual Terminal Path (QTVOPNVT) API.

Write information
INPUT; CHAR(10)

Information about the write operation. The information given in each character is as follows:

1 Key flag. Valid values are:

- *0* The Enter key was pressed.
- *1* The System Request key was pressed. The next read operation returns the System Request menu.
- *2* The Attention key was pressed. In this case, the number of bytes to write must be 0.
- *3* The Test Request key was pressed.
- *4* The Help-in-Error key was pressed.

2 Operation code. This parameter describes the type of write operation to perform. Valid values and their meanings are:

- *blank* Put/get
- *2* Output only
- *3* Put/get
- *4* Save display
- *A* Cancel invite

For detailed descriptions of these codes, see "Write Operation Codes" on page 17.

3 Data stream output error. The negative response code, for example X'10030101', is sent as data in the data buffer.

- *blank* 5250 data in data buffer
- *0* 5250 data in data buffer
- *1* Data stream error; SNA response code data is in the data buffer.

4-10 Reserved. These characters must be blank.

Data buffer
INPUT; CHAR(*)

The server program's buffer containing the data to send to the virtual terminal.

The QTVWRTVT API does not lock the data buffer. Thus, other applications should not use the buffer while the API is using it.

Number of bytes to write
INPUT; BINARY(4)

The number of bytes to write. This number must be smaller than or equal to the size of the data buffer. Valid range of numbers is 0 through 24KB. This parameter must be 0 if character 1 of the write information parameter is 2.

Some write operations do not write data.

Error code
I/O; CHAR(*)

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

Write Operation Codes

The following table describes the operation codes that can be used for the write information parameter.

Write Operation Codes		
Value	**Name**	**Description**
blank	Put/get	Data is being sent to the virtual terminal. The virtual terminal server program is ready for input.
2	Output only	This write operation is in response to a read request that returned an output-only read operation code.
3	Put/get	See the description above.
4	Save display	This write operation is in response to a read request that returned a save display read operation code. No data is associated with this write operation; thus, the data buffer length must be set to 0.
A	Cancel invite	This write operation is in response to a read request that returned a cancel invite read operation code. No data is associated with this write operation; thus, the data buffer length must be set to 0.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF87D4 E	Data sent exceeded the corresponding I/O request.
CPF87F2 E	Virtual terminal handle &1 not valid.
CPF87F3 E	Data buffer length &1 not valid.
CPF87F4 E	Key flag &1 not valid.
CPF87F5 E	Operation code response &1 not valid.
CPF87F7 E	Parameter value &1 not valid.
CPF87F8 E	Unexpected internal system error occurred in program &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Top | "Virtual Terminal APIs," on page 1 | APIs by category

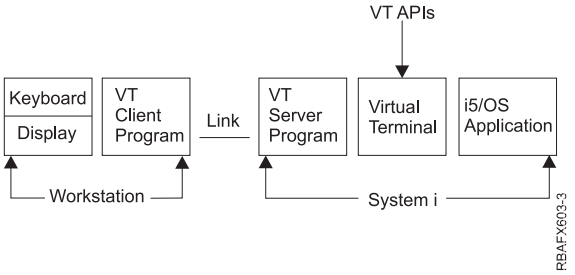
Concepts

These are the concepts for this category.

Distributed 5250 Emulation Model

The Virtual Terminal Client/Server Model example shows a model for a distributed systems environment between a workstation and a System i™ platform. The client program resides on the workstation, while the server program resides on the System i platform. This model is similar to the way the system supports a System i Access workstation function.

Virtual Terminal Client/Server Model Example



The client program running on the workstation shown in the example does the following:

- Accepts data from the server program and displays the data on the workstation display
- Accepts data from the workstation keyboard
- Converts the data from the format required by the workstation display and keyboard to the format required by the server program (5250 data stream)
- Sends the data to the server program

The link between the client program and server program uses DOS and i5/OS® communications support. This may be LU 6.2, TCP/IP, or some other communications protocol.

You can write a server application program in any i5/OS-supported high-level language (HLL), such as the ILE C programming language. The server program provides workstation support for the system acting as the server. The virtual terminal APIs allow a server program to read and write virtual terminal data. Virtual terminal data is always in a 5250 data stream format. See the *IBM® 5494 Functions Reference* manual, SC30-3533, for more information about 5250 data streams. This manual can be viewed online through IBM Publications Center .

The virtual terminal APIs provide an interface between the server program and the virtual terminal. The virtual terminal represents the link between the server program and the i5/OS application and is managed entirely by the i5/OS operating system.

The i5/OS application can be a licensed program or a user-written application that performs a standard data transfer to a workstation. This application can be written in any i5/OS-supported HLL.

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Job Information

Several jobs are involved when you use the virtual terminal APIs. The jobs can be classified into the following groups:

- Server program jobs
- Application jobs

Each group can consist of one or more jobs, depending on the way the server program is written and the way the i5/OS® applications are being run.

The server program can be written to run in a single job or in more than one job, depending on the number of workstations to be supported per job. For example, you can support each workstation using a single job by routing all requests from the workstation client program to a particular server program job.

"Virtual Terminal APIs," on page 1 | APIs by category

Subsystem Information

A server program should run in the same subsystem that other server programs are running. For controlling resource use, such as main storage, a separate subsystem for running all server programs is usually best. This is also advantageous for allowing performance tuning, such as the number of page faults. Generally, i5/OS® applications run in subsystem QBASE.

If the subsystem under which you want the server program to run was created with the system defaults, you will not have to add a workstation entry to the subsystem description. If you do need to add a workstation entry to the subsystem description, however, you can use the Add Work Station Entry (ADDWSE) command.

Before a workstation is allowed to sign on, it must be defined to a subsystem. In this case, the workstation is the virtual terminal device (QPADEVnnnn) that is automatically created by the i5/OS licensed program. The workstation name, workstation type, or *ALL must be specified in the subsystem description. Use the Display Subsystem Description (DSPSBSD) command to see the list of workstation entries defined for a subsystem. The following command can be used to add all workstation types to a subsystem named QBASE:

```
ADDWSE SBSD(QBASE) WRKSTNTYPE(*ALL)
```

For more information about automatically creating virtual terminals, see “Step 1: Setting the Number of Automatically Created Virtual Terminals” on page 20.

Note: The ADDWSE command is valid only when the subsystem description is not active.

“Virtual Terminal APIs,” on page 1 | APIs by category

Data Queues

The i5/OS® operating system uses data queues to send data to the server program. The server program also uses data queues for interprocess communications with other i5/OS applications and API calls.

The data queue must exist before your application uses the virtual terminal APIs to open a path to a virtual terminal. See the Control language topic collection for details about creating and deleting data queues. See “Open Virtual Terminal Path (QTVOPNVT) API” on page 2 for information about how a server program specifies the name of the data queue to be used.

The operating system communicates special events to the server program using the data queue. The system provides information about the special events using a data queue entry.

The following events result in data queue entries being sent:

- The virtual terminal receives data from an i5/OS application
- The operating system closes the virtual terminal (This occurs when an application’s job is canceled.)

The following table shows the structure of i5/OS data queue entries that are sent to the data queue when an application uses the virtual terminal APIs. All data queue entries have the same format.

Format for i5/OS Data Queue Entries		
Name	**Type**	**Description**
Entry Type	CHAR(10)	Always set by i5/OS to *VRTTRM.

Format for i5/OS Data Queue Entries		
Name	**Type**	**Description**
Entry ID	CHAR(2)	Entry ID associated with entry. Valid values for the 2 characters in this parameter are: 1. The operating system is closing (terminating) the session with the virtual terminal. The server program should perform a close to indicate that the server program is done using the virtual terminal. 2. An i5/OS application has sent data to the virtual terminal. See "Read from Virtual Terminal (QTVRDVT) API" on page 11 for information about how to read the data. **Note:** All other values are reserved by the system.
VTHandle	CHAR(16)	The virtual terminal handle associated with the virtual terminal communications path previously opened by calling the Open Virtual Terminal Path (QTVOPNVT) API. See "Open Virtual Terminal Path (QTVOPNVT) API" on page 2 for additional details.
	CHAR(52)	Reserved
Key	CHAR(*)	Key value specified when the virtual terminal communications path was opened. See "Open Virtual Terminal Path (QTVOPNVT) API" on page 2 for additional details on specifying the key value. The key value can be used for retrieving data queues by key.

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Preparing to Use the Virtual Terminal APIs

The following steps are required to prepare your System i™ platform to run an application using the virtual terminal APIs:

1. Set the number of automatically created virtual terminals using the Automatic virtual device configuration indicator (QAUTOVRT) system value
2. Set the Limit security officer device access "Step 2: Setting the Limit Security Officer (QLMTSECOFR) Value" on page 21 system value
3. Create user profiles using the Create User Profile "Step 3: Creating User Profiles" on page 22 command

Step 1: Setting the Number of Automatically Created Virtual Terminals

The i5/OS® operating system uses virtual terminals to allow a server program to interact with its client by sending and receiving data with i5/OS applications. The operating system will automatically select (and create if necessary) these virtual terminals for you.

The QAUTOVRT system value specifies the maximum number of terminals that will be automatically configured by the system. When you set the QAUTOVRT system value, the operating system automatically configures the required virtual controllers and terminals. Controllers coordinate and control the operation of one or more input/output terminals (such as workstations) and synchronize the operation of such terminals with the operation of the entire system. Use the Change System Value (CHGSYSVAL) command to change the value of the QAUTOVRT system value. For example, entering the following command string changes the number of virtual terminals that can be allocated on a system to 50:

```
CHGSYSVAL SYSVAL(QAUTOVRT) VALUE(50)
```

To determine and set the maximum number of users you want signed on to the system at any time, do the following:

- Set the QAUTOVRT system value to *NOMAX, the maximum value allowed.
- Have your users use the system until you decide that the number of virtual terminals created is sufficient for normal system operation.
- Use the Work with Configuration Status (WRKCFGSTS) command to determine the number of workstations configured.
- Change the QAUTOVRT system value from *NOMAX to the number of virtual terminals you require for normal operation.

If you have never allowed virtual terminals to be configured automatically on your system, the QAUTOVRT system value is 0. As a result, you cannot use the virtual terminal APIs because the operating system is not able to create more workstations than the number specified. If you change the QAUTOVRT system value to 10, the next virtual terminal path opened causes the operating system to create a virtual terminal. This virtual terminal is created because the number of virtual terminals on the controller (0) is less than the number specified in the QAUTOVRT system value (10). Even if you change the specified number to 0 again, the next virtual terminal opened may succeed if a virtual terminal exists that is not being used.

If a virtual terminal does not exist or is in use, the operating system does not create a new virtual terminal because the number of virtual terminals currently existing is greater than or equal to the specified QAUTOVRT system value. When the number of virtual terminals that currently exist is greater than or equal to the QAUTOVRT system value, the message CPF8940, "Cannot automatically select virtual device", is sent to the system operator message queue (QSYSOPR). You must either try again when a virtual terminal description becomes available or increase the QAUTOVRT system value.

The operating system uses the following conventions for naming virtual controllers and workstations:

- Virtual controllers named QPACTL*nn* are used for auto-created virtual terminal descriptions.
- Virtual controllers named QVIRCD*nnnn* are used for named virtual terminal descriptions.
- Virtual terminal descriptions named QPADEV*xxxx* are auto-created devices.
- Named virtual terminal devices may be requested using the virtual terminal APIs. An example of a named virtual terminal device would be NEWYORK001.

Consider the following when you allow the operating system to automatically configure workstations:

- The operating system does not delete virtual terminals, even when the number of workstations attached to virtual controllers exceeds the limit set by QAUTOVRT.

  If you want the extra workstations deleted, you must manually delete them.
- The operating system allows a maximum of 254 virtual terminals on the QPACTL01 controller before it creates QPACTL02. This value is usually adequate. If you delete workstations to enforce a smaller value for the QAUTOVRT limit, begin by deleting the workstations from the controller with the highest numeric value in its name (where *nn* in the QPACTL*nn* name is largest).

  Note: Changing this system value affects other products and programs that require automatic configuration. This includes TCP/IP TELNET, 5250 display station pass-through, and any other programs using the virtual terminal APIs.

Step 2: Setting the Limit Security Officer (QLMTSECOFR) Value

The Limit security officer device access (QLMTSECOFR) system value, limits the devices the security officer can sign on to. The security officer controls all of the security authorizations provided by the system. If the QLMTSECOFR value is greater than zero, the security officer must be authorized to use the virtual device descriptions. When this value equals 0, however, the system does not limit the devices the security officer can use to sign on the system.

When the system security level (QSECURITY) system value is set to 30, a security officer with all object authority (*ALLOBJ) must be authorized to use the workstations. For example, for each display station that a security officer wants to sign on to (local, remote, or virtual), the user must authorize the security officer using the following Grant Object Authority (GRTOBJAUT) command:

```
GRTOBJAUT OBJ(display-name) OBJTYPE(*DEVD) AUT(*CHANGE) USER(QSECOFR)
```

This procedure is very important because using the virtual terminal APIs automatically configures virtual terminals (devices). Automatic configuration is a function that names and creates the descriptions of network devices and controllers attached to a line. If the QLMTSECOFR value is set to 0, all virtual terminals automatically configured when you use the virtual terminal APIs can be used by the security officer. If you set the QLMTSECOFR value to 1, your security officer is not able to use the virtual terminals unless you specifically grant object authority to the security officer for that virtual terminal. The automatic configuration support can delete and re-create the virtual terminal. If this occurs, authority must be granted to the security officer each time the virtual terminal is created.

Security Considerations

The number of sign-on attempts allowed increases if virtual terminals are automatically configured. The number of sign-on attempts is equal to the number of system sign-on attempts allowed multiplied by the number of virtual terminals that can be created. The number of system sign-on attempts allowed is defined by the QMAXSIGN system value. The number of virtual terminals that can be created is defined by the QAUTOVRT system value.

Step 3: Creating User Profiles

You should create one or more user profiles on the system for users of the virtual terminal supported by the client and server programs. The default user profile is *SYS. The following example shows a sample user profile:

```
CRTUSRPRF  USRPRF(CLERK1)  PASSWORD(unique-password)
           JOBD(CLERKLIB/CLERKL1)
           TEXT('User profile for one group of clerks')
```

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Creating Your Own Virtual Controllers and Devices

You can create your own virtual controllers and devices (terminals); however, you must use the same naming conventions as the automatic controller and device creation support. You may want to create the virtual terminal descriptions to control the number of sign-on attempts possible by not allowing automatic configuration of virtual terminals (which allows additional sign-on attempts to occur). See "Security Considerations" for additional information.

If you do not want to use automatically created descriptions, do the following:

- Use the Create Controller Description (Virtual Work Station) (CRTCTLVWS) command to create a controller description for a virtual terminal.

  ```
  CRTCTLVWS CTLD(QPACTL01)
            TEXT('Virtual Controller for virtual terminals')
  ```

  Note: You must use the i5/OS® naming convention, QPACTL*nn*, for naming virtual controllers, where *nn* is a decimal number starting at 01.
- Use the Create Device Description (Display) (CRTDEVDSP) command to create a virtual terminal as follows:

  ```
  CRTDEVDSP DEVD(QPADEV0001) DEVCLS(*VRT)
            TYPE(5251) MODEL(11) CTL(QPACTL01)
            TEXT('24 X 80 Monochrome
               Display for Server Program')
  ```

- The i5/OS operating system automatically varies on the controller and terminal that you have created. You must use the i5/OS naming convention, QPADEV*xxxx*, for naming virtual device descriptions, where *xxxx* are alphanumeric characters from 0000 to ZZZZ.

After creating the descriptions, you must authorize the server program to use them. Use the Grant Object Authority (GRTOBJAUT) command to authorize the user profile used by the server program to the descriptions. This can be done using the following commands:

```
GRTOBJAUT OBJ(QPACTL01) OBJTYPE(*CTLD)
          AUT(*CHANGE) USER(user-ID)
GRTOBJAUT OBJ(QPADEV0001) OBJTYPE(*DEVD)
          AUT(*CHANGE) USER(user-ID)
```

You may want to prevent virtual terminals from being created automatically. To do this, set the QAUTOVRT system value to 0 as follows:

```
CHGSYSVAL SYSVAL(QAUTOVRT) VALUE(0)
```

See "Step 1: Setting the Number of Automatically Created Virtual Terminals" on page 20 for additional information.

Note: Changing this system value affects other products and programs that require automatic configuration. This includes TELNET, 5250 display station pass-through, and any other programs using the virtual terminal APIs.

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Developing Client and Server Programs

When developing client and server programs using the virtual terminal APIs, you need to take the following items into consideration:

- The client program should be able to:
  - Interrupt the server program
  - Check the server program's status
  - Discard data from the i5/OS® application
- The user should be able to configure a time-out to be used by the client program while waiting for screens from the server program.
- Pressing the Print key on the workstation should create a file to be printed at either the workstation or the printer on the system.

"Virtual Terminal APIs," on page 1 | APIs by category

Virtual Terminal Runtime Example

To help understand how virtual terminal APIs are used, the following example shows how the i5/OS® operating system, server program, client program, and workstation device (display and keyboard) interact when processing a system request.

This example starts with the server program waiting for a response from the client program, which is waiting for data from the user (keyboard).

1. System request processing starts when you press the appropriate System Request key on the workstation keyboard.
2. The client program informs the server program that the System Request key has been pressed. The protocol used in this case is unique to the particular implementation of these two programs.

3. The server program calls the Write to Virtual Terminal (QTVWRTVT) API for a write request. The flag for the System Request key must be set for this write request. No data is sent to the virtual terminal at this time.
4. The operating system creates a data queue entry for informing the server program that data is available to be read.
5. The server program removes the entry from the data queue by calling the Receive Data Queue (QRCVDTAQ) API and then calls the Read from Virtual Terminal (QTVRDVT) API for a read request. The Cancel Invite operation code is returned. No data is received from the virtual terminal at this time.

   To prevent the client program from sending anymore data (screens), the server program informs the client program that it is no longer receiving data from the client program.

   The server program calls the QTVWRTVT API for a write request. The Operation Code parameter is set to Cancel Invite. No data is sent to the virtual terminal at this time.
6. The operating system creates a data queue entry for informing the server program that data is available to be read.
7. The server program removes the entry from the data queue by calling the QRCVDTAQ API and then calls the QTVRDVT API for a read request. A Save Screen operation code is returned. No data is received from the virtual terminal at this time.

   The server program gets the current screen. This may require requesting the current screen from the client program.

   The server program calls the QTVWRTVT API for a write request, sending the current screen to the virtual terminal. The Operation Code parameter must be set to Save Screen.
8. The operating system creates a data queue entry for informing the server program that data is available to be read.
9. The server program removes the entry from the data queue by calling the QRCVDTAQ API and then calls the QTVRDVT API for a read request. A Put/Get operation code is returned. The data read will be the actual System Request menu.

   The server program sends this data to the client program and waits for a response.
10. The client program updates the display with the System Request menu and waits for a response from the user. The resulting response is sent to the server program.
11. The response is received from the client program, and the server program calls the QTVWRTVT API for a write request, sending the response to the virtual terminal.

    Note: What happens at this point depends on the response to the System Request menu. Additional data may be received from and sent to the virtual terminal. After the response is processed, the following steps occur.
12. The operating system creates a data queue entry for informing the server program that data is available to be read.
13. The server program removes the entry from the data queue by calling the QRCVDTAQ API and then performs a call to the QTVRDVT API for a read request. A Put operation code is returned. The data read is the saved (current) screen that was previously written by the server program to the virtual terminal.

    The server program sends the saved screen to the client program but does not wait for a response.
14. The client program updates the workstation display with the saved screen.
15. The operating system creates a data queue entry for informing the server program that data is available to be read.
16. The server program removes the entry from the data queue by calling the QRCVDTAQ API. An Invite operation code is returned. Note that no data is received from the virtual terminal at this time.
17. The client program is once again waiting for user data, and the server program is waiting for data from the client program.

Top | "Virtual Terminal APIs," on page 1 | APIs by category

Appendix. Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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