Work Station Support APIs (V5R2)

Table of Contents

Work Station Support APIs

The work station support APIs allow you to control the type-ahead characteristics of a work station and to retrieve information about the last output operation to the requester device for the specified interactive job.

Type-ahead, also called **keyboard buffering**, lets the user type data faster than it can be sent to the system. **Attention key buffering** determines how to process the action of pressing an attention key. If attention key buffering is on, the attention key is treated as any other key. If attention key buffering is not on, pressing the attention key results in sending the information to the system even when other work station input is inhibited.

You can enter the parameters for the QWSQRYWS and QWSSETWS APIs in mixed case. The APIs convert them to uppercase. For all other APIs, you must enter all parameters in uppercase.

The keyboard buffering data stream is supported by the following controllers:

- ASCII Work Station Input/Output Processor
- Twinaxial Work Station Input/Output Processor
- 5394 Remote Control Unit
- 5494 Control Unit

IBM Personal Computer Systems attached by iSeries Access or work station emulation (WSE) do not support the type-ahead data stream. Keyboard buffering for these devices is controlled through the emulation programs.

The work station support APIs are:

- Query Keyboard Buffering (QWSQRYWS) determines the current type-ahead and attention key buffering settings.
- Retrieve Output Information (QWSRTVOI) gives the caller information on the last attempted output operation to the requester device for the specified job.
- Set Keyboard Buffering (QWSSETWS) controls the use of the type-ahead and attention key buffering functions.
- Suspend or Restore Display File (QWSSPRST) gives the caller the ability to either suspend the active display file on the requester device or restore a suspended file to the requester device.

Top | APIs by category

Query Keyboard Buffering (QWSQRYWS)API

Required Parameter Group:

1	Keyboard buffering	Output	Char(1)

Optional Parameter Group:

2	Device name	Input	Char(10)

Default Public Authority: *USE

Threadsafe: No

The Query Keyboard Buffering (QWSQRYWS) API sends the data stream command to query the current value for keyboard buffering for a specified display.

Authorities and Locks

Device Authority
    *USE
Display File QDWSTYPA Authority
    *USE

Required Parameter Group

Keyboard buffering
OUTPUT; CHAR(1)

The current setting for keyboard buffering for the device. Valid values are:

0	The type-ahead and attention key buffering functions are off.
1	The type-ahead and attention key buffering functions are on.
2	The type-ahead function is on, and the attention key buffering function is off.

Optional Parameter

Device name

INPUT; CHAR(10)

The display to query for the keyboard buffering value. You can specify a particular device or use this special value:

**REQUESTER*	The job's requesting program device is queried for the keyboard buffering value. This is the default.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF94FC E	Type-ahead data stream not supported by controller.
CPF94FD E	Type-ahead option parameter value not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Retrieve Output Information (QWSRTVOI) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Qualified job name	Input	Char(26)
5	Internal job identifier	Input	Char(16)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve Output Information (QWSRTVOI) API gives the caller information on either the last active record format involved in an output operation to the requester device for the specified job or all the active record formats currently displayed on the requester device for the specified job.

Authorities and Locks

Job Authority

*JOBCTL if the job for which information is retrieved has a different user profile from that of the job that calls the QWSRTVOI API.

Device Authority

*READ authority to the display device that the interactive job is using as the *REQUESTER device.

File Authority

*READ authority to the display file that is active currently.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

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

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. If this value is larger than the actual size of the receiver variable, the result may not be predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The content and format of the information returned for the job. The following format names can be specified:

OINF0100	Display file and last record format name involved in an output operation.
OINF0200	Display file and a list of active records currently on the display.

For more information, see OINF0100 Format or OINF0200 Format.

Qualified job name

INPUT; CHAR(26)

The name of the job for which information is to be returned. The qualified job name has three parts:

Job name	CHAR(10). A specific job name or one of the following special values:
	* — The job in which this program is running. The rest of the qualified job name parameter must be blank.
	**INT* — The internal job identifier locates the job. The user name and job number must be blank.
User name	CHAR(10). A specific user profile name, or blanks when the job name is a special value or *INT.
Job number	CHAR(6). A specific job number, or blanks when the job name specified is a special value or *INT.

Internal job identifier

INPUT; CHAR(16)

The internal identifier for the job. The List Job (QUSLJOB) API creates this identifier. If you do not specify *INT for the job name parameter, this parameter must contain blanks. With this parameter, the system can locate the job more quickly than with a job name.

Error code

I/O; CHAR(*)

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

OINF0100 Format

Offset		Type	Field
Dec	Hex		

0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Display file name
18	12	CHAR(10)	Display file library name
28	1C	CHAR(10)	Display file record format name

OINF0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Offset to the list of active record formats
12	C	BINARY(4)	Number of active record formats
16	10	BINARY(4)	Size of an active record format entry
20	14	CHAR(10)	Display file name
30	1E	CHAR(10)	Display file library name
These fields repeat in the order listed for the number of active record formats		CHAR(10)	Active record format name
		CHAR(10)	Record format type
		BINARY(4)	Start row
		BINARY(4)	End row
		BINARY(4)	Window start column
		BINARY(4)	Window end column

Field Descriptions

Active record format name. The name of an active record format for the active file on the requester device for the specified interactive job. An active record format is a record format that is currently displayed on the requester device.

The following internal record names may be returned:

**ERRSFL*	The name for an internal sub-file created when the ERRSFL keyword is used at the file level in the DDS. The start and end row values both indicate the message line. If the MSGLOC keyword is used in the DDS, the values indicate that line. Otherwise, they indicate the default message line.
**ERRCTL*	The name of an internal sub-file control record created for the ERRSFL keyword.

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.

Display file library name. The name of the library that contains the display file.

Display file name. The name of the display file. The user interface manager (UIM) panel groups generate a user-defined data stream through a display file. The name of the display file (such as QDUI80 or QDUI132) is returned, not the panel group. If format OINF0200 is being used, the record format type for a UIM panel is ***USRDFN**.

Display file record format name. The name of the record format used from the display file.

End row. Ending row of the record format.

Number of active record formats. The number of currently active record formats for the given file. This number is used to determine the size of the list of active record formats array. The maximum number of active records is 25.

Offset to the list of active record formats. An offset from the beginning of format OINF0200 to the list of active record formats.

Record format type. The type of the record format. Blanks are returned in this field if the record format has no special type. Otherwise, one of the following is returned:

**SFL*	Subfile record
**SFLCTL*	Subfile control record
**SFLMSGRCD*	Subfile message record
**MNUBAR*	Menubar record
**PULLDOWN*	Pulldown record
**USRDFN*	The record is a user defined data stream.
**WINDOW*	Window definition record.

If a window is active on the display, any previously written records are inactive. The window definition record is active, as are any records written "in" the window. Non-window definition records active in the window are records with the WINDOW keyword with a parameter that names the window definition record format; these are window reference records. Reported for an active window are:

- An entry that names the window definition record and for which the the start row, end row, start column and end column define the positions of the window's borders on the display. This is the only entry which has a format type of "*WINDOW".

- Another entry with the same record name, if the window definition record contains fields. Only the start row and end row are non-zero in this entry. The start row and end row are the actual rows on the display that the record's fields occupy.

- Any additional records that are active in the window, with their actual start row and end row positions.

If a window is active, the sub-file for the ERRSFL keyword (if present in the DDS) is not active.

Size of an active record format entry. The size of one of the repeating entries in the active records format information.

Start row. Starting row of the record format.

Window end column. If the record format type is *WINDOW, this is the ending column of the window record. For all other record format types, this field is zero.

Window start column. If the record format type is *WINDOW, this is the starting column of the window record. For all other record format types, this field is zero.

Error Messages

Message ID	Error Message Text
CPF2207 E	Not authorized to requester device or display file.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C51 E	Internal job identifier not valid.
CPF3C52 E	Internal job identifier no longer valid.
CPF3C53 E	Job &3/&2/&1 not found.
CPF3C55 E	Job &3/&2/&1 does not exist.
CPF3C57 E	Not authorized to retrieve job information.
CPF3C58 E	Job name specified is not valid.
CPF3C59 E	Internal identifier is not blanks and job name is not *INT.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF919F E	Job is not interactive.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R6

Set Keyboard Buffering(QWSSETWS)API

Required Parameter Group:			
1	Keyboard buffering	Input	Char(1)
Optional Parameter Group:			
2	Device name	Input	Char(10)
Default Public Authority: *USE			
Threadsafe: No			

The Set Keyboard Buffering (QWSSETWS) API controls the type-ahead and attention key buffering functions for a display. With the QWSSETWS API, you can:

- Turn both functions off
- Turn both functions on
- Turn on the type-ahead function without buffering the Attention ke y
- Send the data stream to a specific device

Any changes to the keyboard buffering value made through this program take effect immediately.

The keyboard buffering data stream is supported by the ASCII Work Station Input/Output Processor, Twinaxial Work Station Input/Output Processor, 5394 Remote Control Unit, and 5494 Control Unit. Keyboard buffering for personal computer systems attached using iSeries Access or work station emulation (WSE) is controlled through the emulation programs; these devices do not support the type-ahead data stream.

Authorities and Locks

Device Authority

*USE

Display File QDWSTYPA Authority

*USE

Required Parameter Group

Keyboard buffering

INPUT; CHAR(1)

The setting for keyboard buffering for a display. Valid values are:

0	The type-ahead and attention key buffering functions are off.
1	The type-ahead and attention key buffering functions are on.
2	The type-ahead function is on, and the attention key buffering function is off.

Optional Parameter

Device name

INPUT; CHAR(10)

The device to set the keyboard buffering value on. You can specify the name of a particular device or use this special value:

**REQUESTER*	The keyboard buffering value is set on the job's requesting program device. This is the default.

Error Messages

Message ID	**Error Message Text**
CPF3C90 E	Literal value cannot be changed.
CPF94FC E	Type-ahead data stream not supported by controller.
CPF94FD E	Type-ahead option parameter value not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Suspend or Restore Display File (QWSSPRST) API

Required Parameter Group:

1	Function requested	Input	Char(1)
2	Handle to suspended file	I/O	Binary(4)
3	Screen image option	Input	Char(1)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Suspend or Restore Display File (QWSSPRST) API gives the caller the ability to either suspend the active display file on the requester device or restore a suspended file to the requester device. The suspend function returns a handle to the suspended file. The restore function accepts a handle of a suspended file to be restored.

The QWSSPRST API also allows the caller to have the screen image saved before the file is suspended. When the file is restored, this screen image also can be restored.

Authorities and Locks

None.

Required Parameter Group

Function requested

INPUT; CHAR(1)

The function requested for the start of QWSSPRST. The following functions can be specified:

0	Suspend the active file on the requester device and return a corresponding handle to that file.
1	Restore a suspended file to the requester device. The handle indicates which file will be restored.

Handle to suspended file

I/O; BINARY(4)

When suspending the active file (function 0), a handle to the suspended file will be returned in this parameter. When restoring a suspended file (function 1), this parameter accepts the handle of the file to be restored.

Screen image option

INPUT; CHAR(1)

When suspending the active file (function 0), this parameter gives the caller the option to save the current screen image. When restoring a suspended file (function 1), this parameter gives the caller the option to restore the saved screen image associated with the suspended file. If there is no saved screen image associated with the suspended file, this parameter is ignored when doing a restore.

If the screen image is not saved and restored when the file is restored, it is assumed that the application that is using the file will redraw the screen image. The following options can be specified:

0	If suspending the active file (function 0), do not save the screen image. If restoring an active file (function 1), do not restore the screen image associated with the suspended file.
1	If suspending the active file (function 0), save the current screen image. If restoring an active file (function 1), restore the screen image associated with the suspended file.

This parameter provides a function similar to that of the RSTDSP parameter of the CRTDSPF and CHGDSPF commands. Be aware that the command parameter causes the system to save and restore the screen image if RSTDSP(*YES) is used. The user of this API must request save the screen image at suspend time and restore the screen image at restore time to have the same function.

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.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF4738 E	Parameter value not valid.
CPF5082 E	Suspend request failed.
CPF5083 E	Restore request failed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R5