Work Management APIs (V5R2)

Table of Contents

Work Management APIs

The work management APIs perform functions that are used in a wide variety of applications. These APIs retrieve and manipulate:

- Jobs
- Subsystem storage pools
- Subsystem job queues
- Data areas
- Network attributes
- System status
- System values
- Flight recorders

The work management APIs are:

- [Change Current Job](#) (QWCCCJOB) changes information for the current job.
- [Change Job](#) (QWTCHGJB) changes some of the attributes of a job.
- [Change Pool Attributes](#) (QUSCHGPA) changes the size, activity level, and paging option of system storage pools.
- [Change Pool Tuning Information](#) (QWCCHGTN) changes information about tuning being performed on the system for the different storage pools.
- [Change Subsystem Entry](#) (QWDCSBSE) changes a subsystem entry in the specified subsystem description.
- ≫[Control Thread](#) (QTHMCTLT) holds, releases, or ends the specified thread.≪
- [Control Trace](#) (QWTCTLTR) turns the trace function on and off.
- [Create Job Structures](#) (QWTCTJBS) creates the number of temporary job structures that are passed on the call.
- [Delete Job Structures](#) (QWTDTJBS) deletes the number of temporary job structures that are passed on the call to the API.
- [Dump Flight Recorder](#) (QWTDMPFR) dumps the contents of the flight recorders for jobs that have them.
- [Dump Lock Flight Recorder](#) (QWTDMPLF) dumps the contents of the lock flight recorder for the device that is specified in the parameter that is passed to the program.
- [List Active Subsystems](#) (QWCLASBS) retrieves a list of active subsystems.
- [List Job](#) (QUSLJOB) lists some or all jobs on the system.
- [List Job Schedule Entries](#) (QWCLSCDE) lists the entries in the job schedule QDFTJOBSCD.
- [List Object Locks](#) (QWCLOBJL) generates a list of locks for an object or database file member. An object-level or member-level lock may be specified.
- [List Subsystem Entries](#) (QWDLSBSE) lists some of the different entries in a subsystem description, such as routing entries.
- [List Subsystem Job Queues](#) (QWDLSJBQ) lists the job queues for a subsystem.
- [Move Job](#) (QSPMOVJB) moves jobs from one position to another position within the same job queue or from one job queue to another job queue. Priority and status of the job are affected by the

user's priority level and the status of the target job.

- [Open List of Activation Attributes](#) (QWVOLACT) generates a list of all the activation attributes that are associated with an activation group in a given job.
- [Open List of Activation Group Attributes](#) (QWVOLAGP) generates a list of all the activation groups that are associated with a given job and their attributes.
- [Open List of Job Queues](#) (QSPOLJBQ) generates a list of job queues on the system.
- [Open List of Jobs](#) (QGYOLJOB) generates a list of jobs on the system.
- ≫[Open List of Threads](#) (QWCOLTHD) generates a list of active threads for the job specified in the Job identification parameter.≪
- [Retrieve Call Stack](#) (QWVRCSTK) returns the call stack information for the specified thread.
- [Retrieve Class Information](#) (QWCRCLSI) returns the attributes of a class object.
- [Retrieve Current Attributes](#) (QWCRTVCA) retrieves specific attributes for the current thread.
- [Retrieve Data Area](#) (QWCRDTAA) retrieves the contents of a data area.
- [Retrieve IPL Attributes](#) (QWCRIPLA) returns the settings of options that are used during the IPL.
- [Retrieve Job Description Information](#) (QWDRJOBD) retrieves information from a job description object.
- [Retrieve Job Information](#) (QUSRJOBI) retrieves information, such as job attributes and performance data about a specific job.
- [Retrieve Job Locks](#) (QWCRJBLK) generates a list of objects that have been locked by the job or thread that is specified in the job identification information input parameter.
- [Retrieve Job Queue Information](#) (QSPRJOBQ) retrieves information associated with a specified job queue.
- [Retrieve Job Status](#) (QWCRJBST) returns status and job identification information about the job that is identified by the job identifier parameter.
- ≫[Retrieve Lock Information](#) (QWCRLCKI) generates a list of information about lock holders of the item specified.≪
- ≫[Retrieve Lock Request Information](#) (QWCRLRQI) takes as input a lock request handle that was returned in other APIs and returns information about the program that requested the lock.≪
- ≫[Retrieve Lock Space Attributes](#) (QTRXRLSA) returns information for the specified lock space.≪
- ≫[Retrieve Lock Space Locks](#) (QTRXRLSL) generates a list of objects that have been locked or that have lower level locks held by the specified lock space.≪
- ≫[Retrieve Lock Space Record Locks](#) (QTRXRLRL) lets you generate a list of record locks held by the specified lock space.≪
- [Retrieve Network Attributes](#) (QWCRNETA) retrieves network attributes.
- [Retrieve Profile Exit Programs](#) (QWTRTVPX) retrieves the profile exit flags that have been designated to be called for the specified user ID. The API then places that information into a single variable in the calling program.
- [Retrieve Subsystem Information](#) (QWDRSBSD) retrieves information about a specific subsystem.
- [Retrieve System Status](#) (QWCRSSTS) retrieves a group of statistics that represent the current status of the system.
- [Retrieve System Values](#) (QWCRSVAL) retrieves system values.

- ≫Retrieve Thread Attribute (QWTRTVTA) retrieves job and thread attributes that apply to the job or thread specified in the Job/Thread identification information parameter.≪
- Set Lock Flight Recorder (QWTSETLF) turns the lock flight recorder on and off.
- Set Profile Exit Programs (QWTSETPX) sets for the specified user ID the profile exit programs to call.
- Set Trace (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.

Several of the work management APIs work with job and thread attributes. Those job and thread attributes are discussed in Work Management API Attribute Descriptions.

The work management exit programs are:
- Auxiliary Storage Lower Limit is called when the available storage in the system auxiliary storage pool (ASP) goes below the lower limit.
- Job Notification logs notification messages to data queues when an OS/400 job starts, ends, or is placed on a job queue.
- Power Down System is called when the Power Down System (PWRDWNSYS) or End System (ENDSYS) command is used.
- Preattention Program is called when the user presses the System Attention key.
- Presystem Request Program is called when the user presses the System Request key.
- Trace Job tailors the spooled file output from the Trace Job (TRCJOB) command.

Work Management API Attribute Descriptions

The purpose of this document is to provide one place that describes all the Job and Thread attributes that are used in all the Work Management APIs. The list of attributes encompasses the following APIs:

Retrieve Job Information (QUSRJOBI)

List Job (QUSLJOB)

Retrieve Current Attribute (QWCRTVCA)

Open List of Jobs (QGYOLJOB)

Retrieve Thread Attribute (QWTRTVTA)

Open List of Threads (QWCOLTHD)

Attributes

The following are the list of attributes that have keys associated with them.

Key	Type	Description
0101	CHAR(4)	Active job status
0102	CHAR(1)	Allow multiple threads
0103	CHAR(4)	Active job status for jobs ending
» 0104	See APIs for details	ASP group information«
0201	CHAR(10)	Break message handling
0301	CHAR(1)	Cancel key
0302	BINARY(4)	Coded character set ID
0303	CHAR(2)	Country or region ID
0304	BINARY(4)	Processing unit time used, if less than 2,147,483,647 milliseconds
0305	CHAR(10)	Current user profile
0306	CHAR(1)	Completion status
0307	BINARY(4)	Current system pool identifier
0310	See APIs for details	Current library
0311	CHAR(10)	Character identifier control
0312	BINARY(8), UNSIGNED	Processing unit time used - total for the job
0313	BINARY(8), UNSIGNED	Processing unit time used for database - total for the job
0314	BINARY(4)	Processing unit used - percent used during the elapsed time (job)
0315	BINARY(8), UNSIGNED	Processing unit used - time during the elapsed time (job)
0316	BINARY(4)	Processing unit used for database - percent used during the elapsed time (job)
0317	BINARY(8), UNSIGNED	Processing unit used for database - time during the elapsed time (job)
0318	CHAR(15)	» Client IP address - IPv4«

0319	BINARY(8), UNSIGNED	Processing unit time used - total for the thread
0320	BINARY(8), UNSIGNED	Processing unit time used for database - total for the thread
0321	BINARY(4)	Processing unit used - percent during the elapsed time (thread)
0322	BINARY(8), UNSIGNED	Processing unit used - time during the elapsed time (thread)
0323	BINARY(4)	Processing unit used for database - percent used during the elapsed time (thread)
0324	BINARY(8), UNSIGNED	Processing unit used for database - time during the elapsed time (thread)
» 0326	CHAR(45)	Client IP address - IPv4 or IPv6 «
0401	CHAR(13)	Date and time job became active
0402	CHAR(13)	Date and time job entered system
0403	CHAR(8)	Date and time job is scheduled to run
0404	CHAR(8)	Date and time job was put on this job queue
0405	CHAR(4)	Date format
0406	CHAR(1)	Date separator
0407	CHAR(1)	DBCS-capable
0408	CHAR(10)	DDM conversation handling
0409	BINARY(4)	Default wait
0410	CHAR(13)	Device recovery action
0411	CHAR(10)	Device name
0412	BINARY(4)	Default coded character set identifier
0413	CHAR(1)	Decimal format
0414	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time (job)
0415	BINARY(8), UNSIGNED	Disk I/O count - total for the job
0416	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - asynchronous I/O (job)
0417	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - synchronous I/O (job)
0418	CHAR(13)	Date and time job ended
0419	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time (thread)
0420	BINARY(8), UNSIGNED	Disk I/O count - total for the thread
0421	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - asynchronous I/O (thread)
0422	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - synchronous I/O (thread)
0501	BINARY(4)	End severity
0502	CHAR(1)	End status
0503	CHAR(1)	Exit key

≫ 0504	CHAR(10)	Extended object attribute of entity thread is waiting on≪
0601	CHAR(10)	Function name
0602	CHAR(1)	Function type
0701	CHAR(1)	Signed-on job
0702	CHAR(10)	Group profile name
0703	CHAR(150)	Group profile name - supplemental
0901	CHAR(10)	Inquiry message reply
0902	CHAR(16)	Internal job identifier
0903	CHAR(1)	Initial thread
0904	BINARY(4)	Interactive response time - total during the elapsed time
0905	BINARY(4)	Interactive transactions - count during the elapsed time
1001	CHAR(15)	Job accounting code
1002	CHAR(7)	Job date
1003	CHAR(20)	Job description name - qualified
1004	CHAR(20)	Job queue name - qualified
1005	CHAR(2)	Job queue priority
1006	CHAR(8)	Job switches
1007	CHAR(10)	Job message queue full action
1008	BINARY(4)	Job message queue maximum size
1009	CHAR(26)	Job name
1010	CHAR(1)	Job type
1011	CHAR(1)	Job subtype
1012	CHAR(10)	Job user identity
1013	CHAR(1)	Job user identity setting
1014	BINARY(4)	Job end reason
1015	CHAR(1)	Job log pending
1016	BINARY(4)	Job type - enhanced
1201	CHAR(3)	Language ID
1202	CHAR(1)	Logging level
1203	CHAR(10)	Logging of CL programs
1204	BINARY(4)	Logging severity
1205	CHAR(10)	Logging text
≫ 1206	CHAR(10)	Library of entity thread is waiting on≪
1301	CHAR(8)	Mode name
1302	BINARY(4)	Maximum processing unit time
1303	BINARY(4)	Maximum temporary storage in kilobytes
1304	BINARY(4)	Maximum threads
1305	BINARY(4)	Maximum temporary storage in megabytes
1306	CHAR(10)	Memory pool name
1307	CHAR(1)	Message reply
1401	BINARY(4)	Number of auxiliary I/O requests, if less than 2,147,483,647
1402	BINARY(4)	Number of interactive transactions

1403	BINARY(4)	Number of database lock waits
1404	BINARY(4)	Number of internal machine lock waits
1405	BINARY(4)	Number of nondatabase lock waits
1406	BINARY(8), UNSIGNED	Number of auxiliary I/O requests
» 1407	CHAR(30)	Name of entity thread is waiting on«
1501	CHAR(20)	Output queue name - qualified
1502	CHAR(2)	Output queue priority
» 1503	CHAR(10)	Object type of entity thread is waiting on«
1601	CHAR(10)	Print key format
1602	CHAR(30)	Print text
1603	CHAR(10)	Printer device name
1604	CHAR(10)	Purge
1605	BINARY(4)	Product return code
1606	BINARY(4)	Program return code
1607	CHAR(8)	Pending signal set
1608	BINARY(4)	Process ID number
1609	BINARY(8), UNSIGNED	Page fault count during the elapsed time (job)
1610	BINARY(8), UNSIGNED	Page fault count during the elapsed time (thread)
1660	See APIs for details	Product libraries
1801	BINARY(4)	Response time total
1802	BINARY(4)	Run priority (job)
1803	CHAR(80)	Routing data
1804	BINARY(4)	Run priority (thread)
1901	CHAR(20)	Sort sequence table - qualified
1902	CHAR(10)	Status message handling
1903	CHAR(10)	Status of job on the job queue
1904	CHAR(26)	Submitter's job name - qualified
1905	CHAR(20)	Submitter's message queue name - qualified
1906	CHAR(20)	Subsystem description name - qualified
1907	BINARY(4)	System pool identifier
1908	CHAR(10)	Special environment
1909	CHAR(8)	Signal blocking mask
1910	BINARY(4)	Signal status
1911	CHAR(30)	Server type
1980	See APIs for details	System library list
» 1982	CHAR(10)	Spooled file action«
2001	CHAR(1)	Time separator
2002	BINARY(4)	Time slice
2003	CHAR(10)	Time-slice end pool
2004	BINARY(4)	Temporary storage used in kilobytes

2005	BINARY(4)	Time spent on database lock waits
2006	BINARY(4)	Time spent on internal machine lock waits
2007	BINARY(4)	Time spent on nondatabase lock waits
2008	BINARY(4)	Thread count
2009	BINARY(4)	Temporary storage used in megabytes
2010	CHAR(4)	Thread status
2011	CHAR(1)	Thread type
2012	BINARY(4)	Thread hold count
» 2015	BINARY(4)	Type of entity thread is waiting on «
2101	CHAR(24)	Unit of work ID
2102	BINARY(4)	User return code
2110	See APIs for details	User library list
2702	See APIs for details	All portions of the library list for format RTVC0200
2703	See APIs for details	All portions of the library list for format RTVT0200

Field Descriptions

The following section describes the fields returned in further detail. For details on the thread safety of a particular attribute see the thread safety section in the API being used. The fields are listed in alphabetical order.

Active job status. The active status of the initial thread of the job. Only one status is returned. The possible values are:

no status A blank status field represents a job that is in transition or is not active.

BSCA Waiting in a pool activity level for the completion of an I/O operation to a binary synchronous device.

BSCW Waiting for the completion of an I/O operation to a binary synchronous device.

CMNA Waiting in a pool activity level for the completion of an I/O operation to a communications device.

CMNW Waiting for the completion of an I/O operation to a communications device.

CMTW Waiting for the completion of save-while-active checkpoint processing in another job.

CNDW Waiting on handle-based condition.

CPCW Jobs waiting for the completion of a CPI Communications call.

DEQA Waiting in the pool activity level for completion of a dequeue operation.

DEQW Waiting for completion of a dequeue operation. For example, QSYSARB and subsystem monitors generally wait for work by waiting for a dequeue operation.

DKTA Waiting in a pool activity level for the completion of an I/O operation to a diskette unit.

DKTW Waiting for the completion of an I/O operation to a diskette unit.

DLYW   The Delay Job (DLYJOB) command delays the job for a time interval to end, or for a specific delay end time. The function field shows either the number of seconds the job is to delay (999999), or the specific time when the job is to resume running.

DSC   Disconnected from a work station display.

DSPA   Waiting in a pool activity level for input from a work station display.

DSPW   Waiting for input from a work station display.

END   The job has been ended with the *IMMED option, or its delay time has ended with the *CNTRLD option.

EOFA   Waiting in the activity level to try a read operation again on a database file after the end-of-file has been reached.

EOFW   Waiting to try a read operation again on a database file after the end-of-file has been reached.

EOJ   Ending for a reason other than running the End Job (ENDJOB) or End Subsystem (ENDSBS) command, such as SIGNOFF, End Group Job (ENDGRPJOB), or an exception that is not handled.

EVTW   Waiting for an event. For example, QLUS and SCPF generally wait for work by waiting for an event.

GRP   Suspended by a Transfer Group Job (TFRGRPJOB) command.

HLD   Held.

HLDT   Held due to suspended thread.

ICFA   Waiting in a pool activity level for the completion of an I/O operation to an intersystem communications function file.

ICFW   Waiting for the completion of an I/O operation to an intersystem communications function file.

INEL   Ineligible and not currently in the pool activity level.

JVAA   Waiting in a pool activity level for a Java program operation to complete.

JVAW   Waiting for a Java program operation to complete.

LCKW   Waiting for a lock.

LSPA   Waiting in a pool activity level for a lock space to be attached.

LSPW   Waiting for a lock space to be attached.

MLTA   Waiting in a pool activity level for the completion of an I/O operation to multiple files.

MLTW   Waiting for the completion of an I/O operation to multiple files.

MSGW   Waiting for a message from a message queue.

MTXW   Waiting for a mutex. A **mutex** is a synchronization function that is used to allow multiple jobs or threads to serialize their access to shared data.

MXDW   Waiting for the completion of an I/O operation to a mixed device file.

OPTA	Waiting in a pool activity level for the completion of an I/O operation to an optical device.
OPTW	Waiting for the completion of an I/O operation to an optical device.
OSIW	Jobs waiting for the completion of an OSI Communications Subsystem for OS/400 operation.
PRTA	Waiting in a pool activity level for output to a printer to complete.
PRTW	Waiting for output to a printer to be completed.
PSRW	A prestart job waiting for a program start request.
RUN	Currently running in the pool activity level.
SELW	Waiting for a selection to complete.
SEMW	Waiting for a semaphore. A **semaphore** is a synchronization function that is used to allow multiple jobs or threads to serialize their access to shared data.
SIGS	Stopped as the result of a signal
SIGW	Waiting for a signal
SRQ	The suspended half of a system request job pair.
SVFA	Waiting in a pool activity level for completion of a save file operation.
SVFW	Waiting for completion of a save file operation.
TAPA	The job is waiting in a pool activity level for completion of an I/O operation to a tape device.
TAPW	Waiting for completion of an I/O operation to a tape device.
THDW	Waiting for another thread to complete an operation.
TIMA	Waiting in a pool activity level for a time interval to end.
TIMW	Waiting for a time interval to end.

Active job status for jobs ending. When the active job status field is END or EOJ, this field contains the status of what the initial thread is doing currently. This field is blank if the job is not ending currently. See the active job status field for the list of possible values. For example, the active job status would be EOJ, but the job could be waiting on a lock that could keep the job from ending. The active job status for jobs ending field would then be LCKW.

Allow multiple threads. Whether this job allows multiple user threads. This attribute does not prevent the operating system from creating system threads in the job. Possible values are:

0   This job does not allow multiple user threads.

1   This job allows multiple user threads.

All portions of the library list for format RTVT0200. All portions of the library list will be returned.

≫**ASP group information.** The list of ASP group information for the current thread. This information does not include the system ASP or the basic user ASPs.≪

Break message handling. How this job handles break messages. The possible values are:

NORMAL   The message queue status determines break message handling.

HOLD   The message queue holds break messages until a user or program requests them. The work station user uses the Display Message (DSPMSG) command to display the messages; a program must issue a Receive Message (RCVMSG) command to receive a message and handle it.

NOTIFY   The system notifies the job's message queue when a message arrives. For interactive jobs, the audible alarm sounds if there is one, and the message-waiting light comes on.

Cancel key. Whether the user pressed the cancel key.

0   The user did not press the cancel key.

1   The user did press the cancel key.

   Note: The application or command that was called before this API determines how the key is set.

CCSID of current SQL statement. The CCSID of the current SQL statement string that is returned.

Character identifier control. The character identifier control for the job. This attribute controls the type of CCSID conversion that occurs for display files, printer files, and panel groups. The *CHRIDCTL special value must be specified on the CHRID command parameter on the create, change, or override command for display files, printer files, and panel groups before this attribute will be used.

DEVD   The *DEVD special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

JOBCCSID   The *JOBCCSID special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

≫**Client IP address - IPv4.** The IPv4 address of the client for which this server is doing work. An address is expressed in standard IPv4 dotted-decimal form www.xxx.yyy.zzz (for example, 130.99.128.1). This field is not guaranteed to be an IP address. This field will be blank if the address is not explicitly set to a value by the Change Job (QWTCHGJB) API. For additional usage information of the Client IP address, see the Retrieve Thread Attribute (QWTRTVTA) API.≪

≫**Client IP address - IPv4 or IPv6.** The IPv4 or IPv6 address of the client for which the specified thread of this server most recently communicated with over sockets. If this field is requested for a job, the value from the initial thread of the job will be returned. If a sockets connection has not been established in the initial thread, this field will be blank. An IPv4 address is expressed in standard dotted-decimal form www.xxx.yyy.zzz (for example, 130.99.128.1). An IPv6 address always has at least one occurrence of a colon (':') in the format. Some possible IPv6 address formats would be: ::x (for example, ::1) or ::w.xxx.y.zzz (for example, ::9.130.4.169). For further IPv6 examples and explanation, refer to the Usage Notes section in the Convert IPv4 and IPv6 Addresses Between Text and Binary Form (inet_pton) API. This field is implicitly set by the operating system.≪

Coded character set ID. The coded character set identifier used for this job.

Completion status. The completion status of the job.

blank   The job has not completed.

0   The job completed normally.

1   The job completed abnormally.

Country or region ID. The country or region identifier associated with this job.

Current library. The name of the current library for the thread. If no current library exists, the current library existence field is zero and this field has no entry in the list. If this field is requested for a job, the value for the initial thread of the job will be returned.

Current library existence. The current library existence field:

0   No current library exists.

1   A current library exists.

Current number of pending signals. The number of signals that have been received for a signal monitor but whose signal action has not been taken.

Current SQL statement. The SQL statement that is running currently or was last run in the job.

Current system pool identifier. The identifier of the system-related pool from which main storage is currently being allocated for the job's initial thread. These identifiers are not the same as those specified in the subsystem description, but are the same as the system pool identifiers shown on the system status display. If a thread reaches its time-slice end, the pool the thread is running in can be switched based on the job's time-slice end pool value. The current system pool identifier returned will be the actual pool in which the initial thread of the job is running.

Current user profile. The user profile that the thread for which information is being retrieved is currently running under. This name may differ from the user portion of the job name. If this field is requested for a job, the value for the initial thread of the job will be returned.

Date and time job became active. When the job began to run on the system. This is blank if the job did not become active. It is in the format CYYMMDDHHMMSS, where:

C   Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.

YY   Year

MM   Month

DD   Day

HH   Hour

MM   Minute

SS   Second

Date and time job ended. When the job completed running on the system, in the CYYMMDDHHMMSS format described for the date and time job became active field.

Date and time job entered system. When the job was placed on the system, in the CYYMMDDHHMMSS format described for the date and time job became active field.

Date and time job is scheduled to run. Date and time the job is scheduled to become active. This field is returned as hexadecimal zeros if the job is not a scheduled job. The format for this field is the system time-stamp format.

Date and time job was put on this job queue. This is the date and time this job was put on this job queue. It is in system timestamp format. This field will contain blanks if the job was not on a job queue.

Date format. The format that the date is presented in. The following values are possible:

YMD Year, month, and day format

MDY Month, day, and year format

DMY Day, month, and year format

JUL Julian format (year and day)

Date separator. The value used to separate days, months, and years when presenting a date. The following values are possible:

'/' A slash (/) is used for the date separator.

'-' A dash (-) is used for the date separator.

'.' A period (.) is used for the date separator.

' ' A blank is used for the date separator.

',' A comma (,) is used for the date separator.

DBCS-capable. Whether the job is DBCS-capable.

0 The job is not DBCS-capable.

1 The job is DBCS-capable.

DDM conversation handling. Specifies whether connections using distributed data management (DDM) protocols remain active when they are not being used. The connections include APPC conversations, active TCP/IP connections or Opti-Connect connections. The DDM protocols are used in Distributed Relational Database Architecture(TM) (DRDA(R)) applications, DDM applications, or DB2 Multisystem applications. The following values are possible:

KEEP The system keeps DDM connections active when there are no users, except for the following:

- The routing step ends on the source system. The routing step ends when the job ends or when the job is rerouted to another routing step.
- The Reclaim Distributed Data Management Conversation (RCLDDMCNV) command or the Reclaim Resources (RCLRSC) command runs.
- A communications failure or an internal failure occurs.
- A DRDA connection to an application server not running on the system ends.

DROP The system ends a DDM connection when there are no users. Examples include when an application closes a DDM file, or when a DRDA application runs a SQL DISCONNECT statement.

Decimal format. The decimal format used for this job. The following values are possible:

blank Uses a period for a decimal point, a comma for a 3-digit grouping character, and zero-suppress to the left of the decimal point.

J   Uses a comma for a decimal point and a period for a 3-digit grouping character. The zero-suppression character is in the second position (rather than the first) to the left of the decimal notation. Balances with zero values to the left of the comma are written with one leading zero (0,04). The J entry also overrides any edit codes that might suppress the leading zero.

I   Uses a comma for a decimal point, a period for a 3-digit grouping character, and zero-suppress to the left of the decimal point.

Default coded character set identifier. The default coded character set identifier used for this job. This field contains zeros if the job is not an active job.

Default signal action. The action to be taken by a signal when the signal action specifies that the signal should be handled using the default signal action. Possible values are as follows:

0   End the job

1   Cancel the request

2   Ignore the signal (discard)

3   Stop the job

4   Continue the job if stopped

5   Signal exception

Default wait. The default maximum time (in seconds) that a thread in the job waits for a system instruction, such as a LOCK machine interface (MI) instruction, to acquire a resource. A value of -1 is returned if the value is *NOMAX.

Device name. The name of the device as identified to the system. For an interactive job, the device where the job started; for all other jobs types it will be blanks.

Device recovery action. The action taken for interactive jobs when an I/O error occurs for the job's requesting program device. The possible values are:

*MSG   Signals the I/O error message to the application and lets the application program perform error recovery.

*DSCMSG   Disconnects the job when an I/O error occurs. When the job reconnects, the system sends an error message to the application program, indicating the job has reconnected and that the work station device has recovered.

*DSCENDRQS   Disconnects the job when an I/O error occurs. When the job reconnects, the system sends the End Request (ENDRQS) command to return control to the previous request level.

*ENDJOB   Ends the job when an I/O error occurs. A message is sent to the job's log and to the history log (QHST) indicating the job ended because of a device error.

*ENDJOBNOLIST Ends the job when an I/O error occurs. There is no job log produced for the job. The system sends a message to the QHST log indicating the job ended because of a device error.

Disk I/O count during the elapsed time (job). The number of disk I/O operations performed by the job during the elapsed time. This is the sum of the asynchronous and synchronous disk I/O.

Disk I/O count during the elapsed time (thread). The number of disk I/O operations performed by the specified thread during the elapsed time. This is the sum of the asynchronous and synchronous disk I/O.

Disk I/O count during the elapsed time - asynchronous I/O (job). The number of asynchronous (physical) disk I/O operations performed by the job during the elapsed time. This value is the sum of the asynchronous database and nondatabase reads and writes.

Disk I/O count during the elapsed time - asynchronous I/O (thread). The number of asynchronous (physical) disk I/O operations performed by the specified thread during the elapsed time. This value is the sum of the asynchronous database and nondatabase reads and writes.

Disk I/O count during the elapsed time - synchronous I/O (job). The number of synchronous (physical) disk I/O operations performed by the job during the elapsed time. This value is the sum of the synchronous database and nondatabase reads and writes.

Disk I/O count during the elapsed time - synchronous I/O (thread). The number of synchronous (physical) disk I/O operations performed by the specified thread during the elapsed time. This value is the sum of the synchronous database and nondatabase reads and writes.

Disk I/O count - total for the job. The total number of disk I/O operations performed by the job across all routing steps. This is the sum of the asynchronous and synchronous disk I/O.

Disk I/O count - total for the thread. The total number of disk I/O operations performed by the specified thread. This is the sum of the asynchronous and synchronous disk I/O.

Elapsed time. The time, in milliseconds, that has elapsed between the measurement start time and the current system time. This value is 0 the first time this API is called by this job. The measurement start is set the first time this API is called and when the reset status statistics is set to reset the elapsed time.

End severity. The message severity level of escape messages that can cause a batch job to end. The batch job ends when a request in the batch input stream sends an escape message, whose severity is equal to or greater than this value, to the request processing program.

End status. Whether the system issued a controlled cancelation. The possible values are:

1   The system, the subsystem in which the job is running, or the job itself is canceled.

0   The system, subsystem, or job is not cancelled.

blank   The job is not running.

Exit key. Whether the user pressed the exit key.

0   The user did not press the exit key.

1   The user did press the exit key.

   Note: The application or command that was called before this API determines how the key is set.

≫**Extended object attribute of entity thread is waiting on.** The extended attribute of the object, such as a program or file type, that the thread is waiting on. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG(RPG program) or CLP(CL program), and an object type of *FILE may have a value of PF(physical file), LF(logical file), DSPF(display file), SAVF(save file), and so on. This field may blank if there is no extended attribute associated with the object type, or the type of entity the thread is waiting on is not an OS/400 external object or an OS/400 external object space location. If this field is requested for a job, the value for the initial thread of the job will be returned.≪

Function name. Additional information (as described in the function type field) about the function the initial thread is currently performing. This information is updated only when a command is processed.

Function type. Whether the initial thread is performing a high-level function and what the function type is. The possible values are:

blank The system is not doing a logged function.

C A command is running interactively, or it is in a batch input stream, or it was requested from a system menu. Commands in CL programs or REXX procedures are not logged.

D The initial thread of the job is processing a Delay Job (DLYJOB) command. The function name contains the number of seconds the job is delayed (up to 999999 seconds), or the time when the job is to resume processing (HH:MM:SS), depending on how you specified the command.

G The Transfer Group Job (TFRGRPJOB) command suspended the job. The function name field contains the group job name for that job.

I The initial thread of the job is rebuilding an index (access path). The function name field contains the name of the logical file whose index is rebuilt.

L The system logs history information in a database file. The function name field contains the name of the log (QHST is the only log currently supported).

M The job is a multiple requester terminal (MRT) job if the job type is BATCH and the subtype is MRT, or it is an interactive job attached to an MRT job if the job type is interactive. See job type, subtype, or field for how to determine what type of job this is.

For MRT jobs, the function name field contains information in the following format:

- CHAR(2): The number of requesters currently attached to the MRT job.
- CHAR(1): The field is reserved for a / (slash).
- CHAR(2): The maximum number (MRTMAX) of requesters.
- CHAR(1): Reserved.
- CHAR(3): The never-ending program (NEP) indicator. If an MRT is also an NEP, the MRT stays active even if there are no requesters of the MRT. A value of NEP indicates a never-ending program. A value of blanks indicates that it is not a never-ending program.
- CHAR(1): Reserved.

For interactive jobs attached to an MRT, the function name field contains the name of the MRT procedure.

N The initial thread of the job is currently at a system menu. The function name field contains the name of the menu.

O The job is a subsystem monitor that is performing input/output (I/O) operations to a work station. The function name field contains the name of the work station device to which the subsystem is performing an input/output operation.

P The initial thread of the job is running a program. The function name field contains the name of the program.

R The initial thread of the job is running a procedure. The function name field contains the name of the procedure.

* This does a special function. For this value, the function name field contains one of the following values:

- ADLACTJOB: Auxiliary storage is being allocated for the number of active jobs specified in the QADLACTJ system value. This may indicate that the system value for the initial number of active jobs is too low.
- ADLTOTJOB: Auxiliary storage is being allocated for the number of jobs specified in the QADLTOTJ system value.
- CMDENT: The Command Entry display is being used.
- COMMIT: A commit operation is being performed.
- DIRSHD: Directory shadowing.
- DLTSPLF: The system is deleting a spooled file.
- DUMP: A dump is in process.
- JOBIDXRCY: A damaged job index is being recovered.
- JOBLOG: The system is producing a job log.
- PASSTHRU: The job is a pass-through job.
- RCLSPLSTG: Empty spooled database members are being deleted.
- ROLLBACK: A rollback operation is being performed.
- SPLCLNUP: Spool cleanup is in process.

Group profile name. The name of the group profile that is associated with the initial thread. The user's group authority is used if no specific authority is granted for the user. The value *NONE is returned if no group profile name is found.

Group profile name - supplemental. The name of the group profile associated with the initial thread. The user's group authority is used if no specific authority is granted for the user. Up to 15 supplemental group profile names may be specified. Blanks are returned if no supplemental group profile names are found.

Initial thread. Whether this thread is the initial thread of this job.

0 This thread is not the initial thread.

1 This thread is the initial thread.

Inquiry message reply. How the job answers inquiry messages:

*RQD The job requires an answer for any inquiry messages that occur while this job is running.

*DFT The system uses the default message reply to answer any inquiry messages issued while this job is running. The default reply is either defined in the message description or is the default system reply.

*SYSRPYL The system reply list is checked to see if there is an entry for an inquiry message issued while this job is running. If a match occurs, the system uses the reply value for that entry. If no entry exists for that message, the system uses an inquiry message.

Interactive response time - total during the elapsed time. The total interactive response time for the initial thread, in hundredth of seconds, for the job during the elapsed time. This value does not include the time used by the machine, by the attached input/output (I/O) hardware, and by the transmission lines for sending and receiving data. This field is 0 for noninteractive jobs.

Interactive transactions - count during the elapsed time. The number of user interactions, such as pressing the Enter key or a function key, for the job during the elapsed time for the initial thread. This value is returned for interactive jobs only.

Internal job identifier. A value input to other APIs to increase the speed of locating the job on the system. Only APIs described in this manual use this identifier. The identifier is not valid following an initial program load (IPL). If you attempt to use it after an IPL, an exception occurs.

Job accounting code. An identifier assigned to the job by the system to collect resource use information for the job when job accounting is active.

Job date. This is the date to be used for the job. It is in the format CYYMMDD where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*. This value is for jobs whose status is *JOBQ or *ACTIVE. For jobs with a status of *OUTQ, the value for this field is blank.

   SYSVAL   This job will use the system date.

Job description library name. The library containing the job description.

Job description name. A CHAR(10) representation of the set of job-related attributes used for one or more jobs on the system. These attributes determine how the job is run on the system. Multiple jobs can also use the same job description.

Job description name - qualified. A CHAR(20) representation of the set of job-related attributes used for one or more jobs on the system. These attributes determine how the job is run on the system. Multiple jobs can also use the same job description. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. The data is left-justified and padded with blanks on the right.

Job end reason. The most recent action that caused the job to end. The possible values are:

   0   Job not ending.

   1   Job ending in normal manner.

   2   Job ended while it was still on a job queue.

   3   System ended abnormally.

   4   Job ending normally after a controlled end was requested.

   5   Job ending immediately.

   6   Job ending abnormally.

   7   Job ended due to the CPU limit being exceeded.

   8   Job ended due to the storage limit being exceeded.

   9   Job ended due to the message severity level being exceeded.

   10   Job ended due to the disconnect time interval being exceeded.

   11   Job ended due to the inactivity time interval being exceeded.

   12   Job ended due to a device error.

   13   Job ended due to a signal.

14   Job ended due to an unhandled error.

Job log pending. If the system fails while the job was active or the job ends abnormally, the job log may not be written yet. This flag remains on until the job log has been written. The possible values are:

0   Job log is not pending.

1   Job log is pending and waiting to be written.

Job message queue full action. The action to take when the message queue is full. The values are:

*NOWRAP   When the job message queue is full, do not wrap. This action causes the job to end.

*WRAP   When the job message queue is full, wrap to the beginning and start filling again.

*PRTWRAP   When the job message queue is full, wrap the message queue and print the messages that are being overlaid because of the wrapping.

Job message queue maximum size. The maximum size (in megabytes) that the job message queue can become. The range is 2 to 64.

Job name. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the work station where the job started; for a batch job, you specify the name in the command when you submit the job.

Job number. The system-generated job number.

Job queue library name. The name of the library where the job queue is located.

Job queue name. The CHAR(10) representation of the name of the job queue that the job is currently on, or that the job was on if it is currently active. This value is for jobs whose status is *JOBQ or *ACTIVE. For jobs with a status of *OUTQ, the value for this field is blank.

Job queue name - qualified. The CHAR(20) representation of the name of the job queue that the job is currently on, or that the job was on if it is currently active. This value is for jobs whose status is *JOBQ or *ACTIVE. For jobs with a status of *OUTQ, the value for this field is blank. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. The data is left-justified and padded with blanks on the right.

Job queue priority. The scheduling priority of the job compared to other jobs on the same job queue. The highest priority is 0 and the lowest is 9. This value is for jobs whose status is *JOBQ or *ACTIVE. For jobs with a status of *OUTQ, the value for this field is blank.

Job status. The status of the jobs. The special values are:

*ACTIVE   Active jobs. This includes group jobs, system request jobs, and disconnected jobs.

*JOBQ   Jobs that are currently on job queues.

*OUTQ   Jobs that have completed running but still have output on an output queue.

Job subtype. Additional information about the job type (if any exists). The possible values are:

blank   The job has no special subtype or is not a valid job.

D   The job is an immediate job.

E   The job started with a procedure start request.

F   The job is an AS/400(R) Advanced 36(R) machine server job.

J   The job is a prestart job.

P   The job is a print driver job.

T   The job is a System/36 multiple requester terminal (MRT) job.

U   Alternate spool user.

Job switches. The current setting of the job switches used by this job. This value is returned for all job types.

Job type. The type of job. The possible values for this field are:

blank   The job is not a valid job.

A   The job is an autostart job.

B   The job is a batch job.

I   The job is an interactive job.

M   The job is a subsystem monitor job.

R   The job is a spooled reader job.

S   The job is a system job.

W   The job is a spooled writer job.

X   The job is the SCPF system job.

Refer to [Comparing Job Type and Subtype with the Work with Active Job Command](#) for information about how the job type field and the job subtype field equate to the type field in the Work with Active Job (WRKACTJOB) command.

Job type - enhanced. The type of job. This field combines the job type and job subtype fields. The possible values are:

0110   Autostart job

0210   Batch job

0220   Batch immediate job

0230   Batch - System/36 multiple requester terminal (MRT) job

0240   Batch - alternate spool user

0310   Communications job - procedure start request job

0910   Interactive job

0920   Interactive job - Part of group

0930   Interactive job - Part of system request pair

0940   Interactive job - Part of system request pair and part of a group

1610   Prestart job

1620   Prestart batch job

1630   Prestart communications job

1810   Reader job

1910   Subsystem job

1920   System job (all system jobs including SCPF)

2310   Writer job (including both spool writers and print drivers)

Job user identity. The user profile name by which the job is known to other jobs on the system. The job user identity is used for authorization checks when other jobs on the system attempt to operate against the job. For more detail on how the job user identity is set and used, refer to the Set Job User Identity (QWTSJUID) API in the Security part. For jobs that are on a job queue or have completed running, the job user identity is same as the user name from the qualified job name. This field will return blanks for these jobs. A value of *N is returned if the job user identity is set, but the user profile to which it is set no longer exists.

Job user identity setting. An indicator of the method by which the job user identity was set. Possible values are as follows:

o   The job is currently running single threaded and the job user identity is the name of the user profile under which the job is currently running. This value is also returned for jobs that are on a job queue or have completed running. This has the same meaning as a value of *DEFAULT on the Display Job Status Attributes display.

1   The job user identity was explicitly set by an application using one of the Set Job User Identity APIs, QWTSJUID or QwtSetJuid(). The job may be running either single threaded or multithreaded. This has the same meaning as a value of *APPLICATION on the Display Job Status Attributes display.

2   The job is currently running multithreaded and the job user identity was implicitly set by the system when the job became multithreaded. It was set to the name of the user profile that the job was running under when it became multithreaded. This has the same meaning as a value of *SYSTEM on the Display Job Status Attributes display.

Language ID. The language identifier associated with this job.

Length of current SQL statement. The length of the current SQL statement. Zero indicates that a current SQL statement could not be returned. This can occur if an SQL statement has never been issued in the job, if the job is ending, if the program or package that contained the SQL statement no longer exists, or if the API is unable to access the SQL statement.

Library name. The name of the library object.

»**Library of entity thread is waiting on.** The library name of the entity that the thread is waiting on. This field may be blank if the type of entity the thread is waiting on is not an OS/400 external object, a member object, an OS/400 external object space location, a lock space, or if there is no library associated with the

particular type of entity. If the type of entity the thread is waiting on is a member object, this field will return the library name of the file that contains the member. If this field is requested for a job, the value for the initial thread of the job will be returned.《

Library text description. The text description of the library object. This field is blank if no text description is specified.

》**Lock wait time - time during the elapsed time.** The amount of time (in milliseconds) that the initial thread has to wait to obtain database, nondatabase, and internal machine locks during the elapsed time.《

Logging level. What type of information is logged. The possible values are:

0   No messages are logged.

1   All messages sent to the job's external message queue with a severity greater than or equal to the message logging severity are logged. This includes the indication of job start, job end and job completion status.

2   The following information is logged:

- Level 1 information
- Request messages that result in a high-level message with a severity code greater than or equal to the logging severity cause the request message and all associated messages to be logged.

    Note: A high-level message is one that is sent to the program message queue of the program that receives the request message. For example, QCMD is an IBM-supplied request processing program that receives request messages.

3   The following information is logged:

- Level 1 and 2 information
- All request messages
- Commands run by a CL program are logged if it is allowed by the logging of CL programs job attribute and the log attribute of the CL program.

4   The following information is logged:

- All request messages and all messages with a severity greater than or equal to the message logging severity, including trace messages.
- Commands run by a CL program are logged if it is allowed by the logging of CL programs job attribute and the log attribute of the CL program.

Logging of CL programs. Whether or not commands are logged for CL programs that are run. The possible values are *YES and *NO.

Logging severity. The severity level that is used in conjunction with the logging level to determine which error messages are logged in the job log. The values range from 00 through 99.

Logging text. The level of message text that is written in the job log when a message is logged according to the logging level and logging severity. The possible values are:

*MSG   Only the message text is written to the job log.

*SECLVL   Both the message text and the message help (cause and recovery) of the error message are written to the job log.

NOLIST If the job ends normally, no job log is produced. If the job ends abnormally (the job end code is 20 or higher), a job log is produced. The messages that appear in the job log contain both the message text and the message help.

Maximum number of signals retained. The number of signals that are kept for a signal monitor when the signal action is blocked.

Maximum processing unit time. The maximum processing unit time (in milliseconds) that the job can use. If the job consists of multiple routing steps, this is the maximum processing unit time that the current routing step can use. If the maximum time is exceeded, the job is ended. A value of -1 is returned for *NOMAX. A zero is returned if the job is not active.

Maximum temporary storage in kilobytes. The maximum amount of auxiliary storage (in kilobytes) that the job can use. If the job consists of multiple routing steps, this is the maximum temporary storage that the routing step can use. This temporary storage is used for storage required by the program itself and by implicitly created internal system objects used to support the routing step. (It does not include storage in the QTEMP library.) If the maximum temporary storage is exceeded, the job is ended. This does not apply to the use of permanent storage, which is controlled through the user profile. A value of -1 is returned for *NOMAX.

Maximum temporary storage in megabytes. The maximum amount of auxiliary storage (in megabytes) that the job can use. If the job consists of multiple routing steps, this is the maximum temporary storage that the routing step can use. This temporary storage is used for storage required by the program itself and by implicitly created internal system objects used to support the routing step. (It does not include storage in the QTEMP library.) If the maximum temporary storage is exceeded, the job is ended. This does not apply to the use of permanent storage, which is controlled through the user profile. A value of -1 is returned for *NOMAX.

Maximum threads. The maximum number of threads that a job can run with at any time. If multiple threads are initiated simultaneously, this value may be exceeded. If this maximum value is exceeded, the excess threads will be allowed to run to their normal completion. Initiation of additional threads will be inhibited until the maximum number of threads in the job drops below this maximum value. A value of -1 is returned for *NOMAX.

Note: Depending on the resources used by the threads and the resources available on the system, the initiation of additional threads may be inhibited before this maximum value is reached.

Memory pool name. The name of the memory pool in which the job started running. The name may be a number, in which case it is a private pool associated with a subsystem. The following special values may be returned.

MACHINE	This job is running in the machine pool.
BASE	This job is running in the base system pool, which can be shared with other subsystems.
INTERACT	This job is running in the shared pool used for interactive work.
SPOOL	This job is running in the shared pool for spooled writers.
*SHRPOOL1 - *SHRPOOL60*	This job is running in the identified shared pool.
01 - 99	This job is running in the identified private pool. This value is right-adjusted and padded with blanks.

Message reply. Whether the job is waiting for a reply to a specific message. The field applies only when the active job status or active job status for job ending is MSGW. Possible values are:

0 The job currently is not in message wait status.

1 The job is waiting for a reply to a message.

2 The job is not waiting for a reply to a message.

Mode name. The mode name of the advanced program-to-program communications device that started the job. Possible values are:

*BLANK   The mode name is *BLANK.

blank   The mode name is blanks.

Mode name   The name of the mode.

≫**Name of entity thread is waiting on.** The name of the entity that the thread is waiting on. The format of the name will vary based on the type of entity the thread is waiting on.

For an OS/400 external object, an internal system object, an OS/400 external object space location, or an internal system object space location, the format of the name will be as follows(however, an OS/400 external object and an OS/400 external object space location will only return up to a ten character name):

Offset			
Dec	Hex	Type	Field
0	0	CHAR(30)	Extended name

For a member object, the format of the name will be as follows:

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	File
10	A	CHAR(10)	Member
20	14	CHAR(10)	Reserved

For a lock space object, the format of the name will be as follows:

Offset			
Dec	Hex	Type	Field
0	0	CHAR(20)	Lock space ID
20	14	CHAR(10)	Reserved

This field will be blank if the type of entity the thread is waiting on is unknown or the thread is not waiting. This field will also be blank if the name is for an internal system object or an internal system object space location and the user does not have *JOBCTL special authority. If this field is requested for a job, the value for the initial thread of the job will be returned.≪

Number of auxiliary I/O requests. The number of auxiliary I/O requests performed by the job across all routing steps. This includes both database and nondatabase paging. This is an unsigned BINARY(8) value.

Number of auxiliary I/O requests, if less than 2,147,483,647. The number of auxiliary I/O requests performed by the job across all routing steps. This includes both database and nondatabase paging. If the number of auxiliary I/O requests is greater than or equal to 2,147,483,647, a value of -1 is returned. Use the **Number of auxiliary I/O requests** field to retrieve values that are greater than or equal to 2,147,483,647.

Number of bytes available. All of the available bytes for use in your application. **Note:** When you request format JOBI0700 for the Retrieve Job Information (QUSRJOBI) API, the actual length depends on how many libraries are in the library list.

Number of bytes returned. The number of bytes returned to the user. This may be some but not all of the bytes available.

Number of database lock waits. The number of times that the initial thread had to wait to obtain a database lock. (These performance attributes may be a cumulative job total in a future release.)

Number of interactive transactions. The count of operator interactions, such as pressing the Enter key or a function key. This field is zero for jobs that have no interactions.

Number of internal machine lock waits. The number of times that the initial thread had to wait to obtain an internal machine lock. (These performance attributes may be a cumulative job total in a future release.)

Number of libraries in SYSLIBL. The number of libraries in the system part of the library list of the initial thread.

Number of libraries in USRLIBL. The number of libraries in the user library list of the initial thread.

Number of nondatabase lock waits. The number of times that the initial thread had to wait to obtain a nondatabase lock. (These performance attributes may be a cumulative job total in a future release.)

Number of product libraries. The number of product libraries found in the library list of the initial thread.

Number of signal monitors. The number of signal monitors that are present for the job.

Number of SQL open cursors. The number of SQL cursors that are currently open for the job.

Object library for SQL cursor. The name of the library the object is in that contains the associated SQL open cursor.

Object name for SQL cursor. The name of the object that contains the associated SQL open cursor.

Object type for SQL cursor. The type of object with which the SQL cursor is associated.

»Object type of entity thread is waiting on. The object type of the entity the thread is waiting on. This field may be blank if the type of entity that the thread is waiting on is not a defined OS/400 external object, a member object, an internal system object, an OS/400 external object space location, or an internal system object space location. For a list of all the available OS/400 object types, see the Control Language (CL) topic in the iSeries Information Center. For a list of all internal object types, see the *OS/400 Diagnostics Aids* book. If this field is requested for a job, the value for the initial thread of the job will be returned.«

Offset to current SQL statement. The offset from the start of the format to the start of the current SQL statement.

Offset to SQL open cursor data. The offset from the start of the format to the start of the SQL open cursor data.

Offset to signal monitor data. The offset from the start of the format to the start of the signal monitor data.

Output queue library name. The name of the library containing the output queue.

Output queue name. The name of the default output queue that is used for spooled output produced by this job. The default output queue is only for spooled printer files that specify *JOB for the output queue.

»Output queue name - qualified. The qualified name of the default output queue that is used for spooled output produced by this job and the name of the library that contains the output queue. The default output queue is only for spooled printer files that specify *JOB for the output queue. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. The data is left-justified and padded with blanks on the right.«

Output queue priority. The output priority for spooled output files that this job produces. The highest priority is 0, and the lowest is 9.

Page fault count during the elapsed time (job). The number of times an active program referenced an address that is not in main storage for the current routing step of the specified job during the elapsed time.

Page fault count during the elapsed time (thread). The number of times an active program referenced an address that is not in main storage for the specified thread during the elapsed time.

»Page faults. The number of times an active program referenced an address that is not in main storage during the current routing step of the specified job.«

Pending signal set. A bit field that is used to determine the set of signals that have been received but not acted upon by a signal monitor. The nth bit in the pending signal set represents the nth signal monitor defined for the job. If a bit within the pending signal set has a value of 1, then a signal is present but has not been acted upon yet.

Print key format. Whether border and header information is provided when the Print key is pressed.

NONE   The border and header information is not included with output from the Print key.

PRTBDR   The border information is included with output from the Print key.

PRTHDR   The header information is included with output from the Print key.

PRTALL   The border and header information is included with output from the Print key.

Print text. The line of text (if any) that is printed at the bottom of each page of printed output for the job.

Printer device name. The printer device used for printing output from this job.

Process ID number. A unique UNIX-style process ID number (PID) that is associated with the current routing step of the job. A value of 1 indicates that the PID has not been set.

Processing unit time used for database - total for the job. The amount of processing unit time (in milliseconds) that the job used for processing data base requests across all routing steps. This is an unsigned BINARY(8) value.

Processing unit time used for database - total for the thread. The amount of processing unit time (in milliseconds) that the specified thread used for processing database requests across all routing steps. This is an unsigned BINARY(8) value.

Processing unit time used, if less than 2,147,483,647 milliseconds. The amount of processing unit time (in milliseconds) that the job used. If the processing unit time used is greater than or equal to 2,147,483,647 milliseconds, a value of -1 is returned. Use the **Processing unit time used - total for the job** field to retrieve values that are greater than or equal to 2,147,483,647.

Processing unit time used - total for the job. The amount of processing unit time (in milliseconds) that the job used across all routing steps. This is an unsigned BINARY(8) value.

Processing unit time used - total for the thread. The amount of processing unit time (in milliseconds) used by the specified thread. This is an unsigned BINARY(8) value.

Processing unit used for database - percent used during the elapsed time (job). The percentage of processing unit used for database processing during the elapsed time by the specified job. For multiple-processor systems, this is the average across processors.

Processing unit used for database - percent used during the elapsed time (thread). The percentage of processing unit used for database processing during the elapsed time by the specified thread. For multiple-processor systems, this is the average across processors.

Processing unit used for database - time during the elapsed time (job). The amount of processing unit time (in milliseconds) used for database processing during the elapsed time by the specified job.

Processing unit used for database - time during the elapsed time (thread). The amount of processing unit time (in milliseconds) used for database processing during the elapsed time by the specified thread.

Processing unit used - percent during the elapsed time (job). The percentage of processing time used during the elapsed time. For multiple-processor systems, this is the average across processors.

Processing unit used - percent during the elapsed time (thread). The percentage of processing time used during the elapsed time by the specified thread. For multiple-processor systems, this is the average across processors.

Processing unit used - time during the elapsed time (job). The amount of processing unit time (in milliseconds) used during the elapsed time by the specified job.

Processing unit used - time during the elapsed time (thread). The amount of processing unit time (in milliseconds) used during the elapsed time by the specified thread.

Product libraries. The libraries that contain product information for the thread. If requesting this field for a job, the information for the initial thread of the job will be returned. If this field is defined as a CHAR(11), a blank will be in the last position of the name.

Product return code. The return code set by the compiler for Integrated Language Environment (ILE) languages. Refer to the appropriate ILE-conforming language manual for possible values. This field is scoped to the job and represents the most recent return code set by any thread within the job.

Program return code. If the job contains any RPG, COBOL, data file utility (DFU), or sort utility programs, the completion status of the last program that has finished running is shown; otherwise, a value of zero is shown.

Purge. Whether or not the job is eligible to be moved out of main storage and put into auxiliary storage at the end of a time slice or when it is beginning a long wait (such as waiting for a work station user's response). This attribute is ignored when more than one thread is active within the job. The possible values are:

YES   The job is eligible to be moved out of main storage and put into auxiliary storage. A job with multiple threads, however, is never purged from main storage.

NO   The job is not eligible to be moved out of main storage and put into auxiliary storage. When main storage is needed, however, pages belonging to a thread in the job may be moved to auxiliary storage. Then, when a thread in the job runs again, its pages are returned to main storage as they are needed.

blank   Not used for job types *JOBQ or *OUTQ, or for invalid jobs.

≫Relational Database name. The name used to uniquely identify a data source or relational database.≪

Reserved. An ignored field.

Response time total. The total amount of response time for the initial thread, in milliseconds. This value does not include the time used by the machine, by the attached input/output (I/O) hardware, and by the transmission lines for sending and receiving data. This field is zero for jobs that have no interactions. A value of -1 is returned if the field is not large enough to hold the actual result.

Routing data. The routing data that is used to determine the routing entry that identifies the program to start for the routing step.

Run priority (job). The priority at which the job is currently running, relative to other jobs on the system. The run priority ranges from 0 (highest priority) to 99 (lowest priority).

Run priority (thread). The run priority for the thread relative to the priority of the other threads that are running in the system. This is displayed as a number ranging from 0 (highest priority) to 99 (lowest priority). The value may never be higher than the run priority for the job in which the thread is running.

Server mode for Structured Query Language. Whether or not Structured Query Language (SQL) statements should run in a separate server job. The possible values are:

0   The SQL statements will not run in a separate server job.

1   The SQL statements will run in a separate server job. Each SQL connection will be allowed to run with a different user profile and separate transaction scoping.

Server type. The type of server represented by the job. A value of blanks indicates that the job is not part of a server.

Signal action. The action to be taken when a signal is received for a signal monitor. Possible values are as follows:

-1   The signal associated with this signal monitor is not supported.

0   Handle the signal using the signal default action.

1   Ignore the signal (discard).

2   Handle the signal by running the signal catching function.

Signal blocking mask. A bit field that is used to represent the set of signals whose signal actions are to be held for the initial thread of the job. The nth bit in the signal blocking mask represents the nth signal monitor defined for the initial thread. If a bit within the signal blocking mask has a value of 1, then the signal action is blocked and the signal is held.

Signal monitor data. The signal information for a given signal monitor for the job. This information consists of the signal number, signal action, default signal action, maximum number of signals to be retained, and the current number of signals pending.

Signal number. A numeric value assigned to the signal monitor. This value is used to locate the signal monitor for the job when a signal is sent to the job.

Signal status. A numerical value used to determine if the job is enables to receive signals from another job or the system.

0   The job is not enabled for signals. This job cannot receive signals from another job or system.

1   The job is enabled for signals. This job can receive signals from another job or system.

Signed-on job. Whether the job is a to be treated like a signed-on user on the system.

0   The job should be treated like a signed-on user.

1   The job should not be treated like a signed-on user.

Size of SQL open cursor data. The size of a single entry for a given SQL cursor for the job. This information consists of the object name, object library, object type, SQL cursor name, and SQL statement name.

Sort sequence library. The sort sequence library associated with this job.

Sort sequence table name. The sort sequence table associated with this job. Possible values are:

*HEX   No sort sequence table is used. The hexadecimal values of the characters are used to determine the sort sequence.

*LANGIDSHR   The sort sequence table used can contain the same weight for multiple characters, and it is the shared weight sort table associated with the language specified in the LANGID parameter.

*LANGIDUNQ   The sort sequence table used must contain a unique weight for each character in the code page, and it is the unique weight sort table associated with the language specified in the LANGID parameter.

≫**Sort sequence table - qualified.** The qualified name of the sort sequence table associated with this job. The format of the qualified name is a 10-character sort sequence table name followed by a 10-character library name. The data is left-justified and padded with blanks on the right. If the sort sequence table name is a special value, the library name is blank.≪

Special environment. Whether the job is running in a particular environment. Possible values are:

*NONE   The job is not running in any special environment.

*S36   The job is running in the System/36 environment.

blank   This job is not currently active.

≫**Spooled file action.** Whether spooled files are accessed through job interfaces after the job has completed is normal activity. The possible values are:

*KEEP   The spooled files are kept with the job when the job completes its activity.

*DETACH   The spooled files are detached from the job when the job completes its activity.≪

SQL cursor name. The name of the SQL cursor.

SQL open cursor data. The SQL cursor information for a given SQL cursor for the job. This information consists of the object name, object library, object type, SQL cursor name, and SQL statement name.

SQL statement name. The name of the SQL statement that is associated with the SQL cursor.

Status message handling. Whether you want status messages displayed for this job. The possible values are:

*NONE   This job does not display status messages.

NORMAL   This job displays status messages.

Status of current SQL statement. The status of the current SQL statement. The possible values are:

0   The SQL statement returned is running currently.

1   The SQL statement returned has completed.

Status of job on the job queue. The status of this job on the job queue.

blank   This job was not on a job queue.

SCD   This job will run as scheduled.

HLD   This job is being held on the job queue.

RLS   This job is ready to be selected.

Submitter's job name. The job name of the submitter's job. If the job has no submitter, this field is blank.

≫Submitter's job name - qualified. The qualified job name of the submitter's job. The format of the qualified job name is a 10-character simple job name, a 10-character user name, and a 6-character job number. If the job has no submitter, this field is blank.≪

Submitter's job number. The job number of the submitter's job. If the job has no submitter, this field is blank.

Submitter's message queue library name. The name of the library that contains the message queue. If the job has no submitter, this field is blank.

Submitter's message queue name. The name of the message queue where the system sends a completion message when a batch job ends. If the job has no submitter, this field is blank.

≫Submitter's message queue name - qualified. The qualified name of the message queue where the system sends a completion message when a batch job ends and the name of the library that contains the message queue. If the job has no submitter, this field is blank. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. The data is left-justified and padded with blanks on the right.≪

Submitter's user name. The user name of the submitter. If the job has no submitter, this field is blank.

Subsystem description library name. The library that contains the subsystem description. This value is only for jobs whose status is *ACTIVE. For jobs with a status of *OUTQ or *JOBQ, the value for this field is blank.

Subsystem description name. The name of the subsystem in which an active job is running. This value is only for jobs whose status is *ACTIVE. For jobs with status of *OUTQ or *JOBQ, the value for this field is blank.

≫Subsystem description name - qualified. The qualified name of the subsystem in which an active job is running. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. This value is only for jobs whose status is *ACTIVE. For jobs with status of *OUTQ or *JOBQ, the value for this field is blank.≪

System library list. The system portion of the library list of the thread. If requesting this field for a job, the information for the initial thread of the job will be returned. If this field is defined as a CHAR(11), a blank

will be in the last position of the name.

System pool identifier. The identifier of the system-related pool from which the job's main storage is allocated. These identifiers are not the same as those specified in the subsystem description, but are the same as the system pool identifiers shown on the system status display. This is the pool that the threads in the job start in. Also see the Current system pool identifier field for more information.

Temporary storage used in kilobytes. The amount of auxiliary storage (in kilobytes) that is currently allocated to this job.

Note: This value will reach a maximum of 2 147 483 647 kilobytes. If the actual temporary storage used is larger than that value, this field will return 2 147 483 647 kilobytes. It is recommended that the temporary storage used in megabytes field be used to get over this limit.

Temporary storage used in megabytes. The amount of auxiliary storage (in megabytes) that is currently allocated to this job. This is an unsigned BINARY(4) value.

Thread count. The count of the current number of active threads in the process at the time of the materialization. An **active thread** may be either actively running, suspended, or waiting for a resource.

Thread hold count. The count of the number of times that the specified thread has been held using the hold thread interface. If this field is requested for a job, the value for the initial thread of the job will be returned.

Thread status. The current status of the thread. If this field is requested for a job, the value for the initial thread of the job will be returned. The status of a thread may be one of the following values:

Blank	The status of the thread is unknown.
CMTW	The thread is waiting for the completion of save-while-active checkpoint processing in another job. This wait is necessary to prevent a partial commitment control transaction from being saved to the media.
CNDW	The thread is waiting for a condition.
DEQA	The thread is waiting for completion of a dequeue operation in the pool activity level.
DEQW	The thread is waiting for completion of a dequeue operation. For example, a server may wait for work by waiting for a dequeue operation
EVTW	The thread is waiting for an event.
HLD	The thread is in a job that is being held.
HLDT	The thread is being held.
INEL	The thread is ineligible and not currently in the pool activity level.
JVAA	The thread is waiting for completion of a Java program operation in the pool activity level.
JVAW	The thread is waiting for completion of a Java program operation.
LCKW	The thread is waiting for a lock.
≫*LSPA*	The thread is waiting for a lock space to be attached while in a pool activity level.
LSPW	The thread is waiting for a lock space to be attached.≪

MTXW   The thread is in a mutex wait. A mutex is a synchronization function that is used to allow multiple threads to serialize their access to shared data.

RUN   The thread is currently running in the activity level.

SELW   The thread is in a select wait. More information on the select() function is in the Sockets APIs chapter in the System API Reference, SC41-5801.

SEMW   The thread is waiting for a semaphore. A semaphore is a synchronization function that is used to allow multiple jobs or threads to serialize their access to shared data.

SIGS   The thread has been held by a signal.

SIGW   The thread is waiting for a signal.

THDW   The thread is waiting for another thread to complete an operation.

TIMA   The thread is waiting, in the activity level, for a time interval to end.

TIMW   The thread is waiting for a time interval to end.

Thread type. The thread type indicates how the thread was created. If this field is requested for a job, the value for the initial thread of the job will be returned. The type of a thread may be one of the following values:

0   The thread was created either as the initial thread of the job or explicitly by the application.

1   The thread was created by an operating system function.

Time separator. The value used to separate hours, minutes, and seconds when presenting a time. The following values are possible:

':'   A colon (:) is used for the time separator.

'.'   A period (.) is used for the time separator.

' '   A blank is used for the time separator.

','   A comma (,) is used for the time separator.

Time slice. The maximum amount of processor time (in milliseconds) given to each thread in this job before other threads in this job and in other jobs are given the opportunity to run. The time slice establishes the amount of time needed by a thread in this job to accomplish a meaningful amount of processing. At the end of the time slice, the thread might be put in an inactive state so that other threads can become active in the storage pool. Values retrieved range from 8 through 9999999 (that is, 9 999 999 milliseconds or 9999.999 seconds). Although you can specify a value of less than 8, the system takes a minimum of 8 milliseconds to run a process.

Time-slice end pool. Whether you want a thread in an interactive job moved to another main storage pool at the end of its time slice. The possible values are:

*NONE   A thread in the job does not move to another main storage pool when it reaches the end of its time slice.

*BASE   A thread in the job moves to the base pool when it reaches the end of its time slice.

Time spent on database lock waits. The cumulative amount of time, in milliseconds, that the initial thread has had to wait to obtain database locks. (These performance attributes may be a cumulative job total in a future release.)

Time spent on internal machine lock waits. The cumulative amount of time, in milliseconds, that the initial thread has had to wait to obtain internal machine locks. (These performance attributes may be a cumulative job total in a future release.)

Time spent on nondatabase lock waits. The cumulative amount of time, in milliseconds, that the initial thread has had to wait to obtain nondatabase locks. (These performance attributes may be a cumulative job total in a future release.)

≫**Type of entity thread is waiting on.** The type of entity, such as an OS/400 external object, that the thread is waiting on. If this field is requested for a job, the value for the initial thread of the job will be returned. The type of entity may be one of the following values:

-1   Thread is not waiting

1   OS/400 external object

2   Member object

3   Internal system object

4   OS/400 external object space location

5   Internal system object space location

6   Lock space object

999   Unknown type≪

Unit of work ID. The unit of work ID is used to track jobs across multiple systems. If a job is not associated with a source or target system using advanced program-to-program communications (APPC), this information is not used. Every job on the system is assigned a unit of work ID. The unit-of-work identifier is made up of:

Location name   CHAR(8). The name of the source system that originated the APPC job.

Network ID   CHAR(8). The network name associated with the unit of work.

Instance   CHAR(6). The value that further identifies the source of the job. This is shown as hexadecimal data.

Sequence number   CHAR(2). A value that identifies a checkpoint within the application program.

User library list. The user portion of the library list for the thread. If requesting this field for a job, the information for the initial thread of the job will be returned. If this field is defined as a CHAR(11), a blank will be in the last position of the name.

User name. The user name of the job, which is the same as the name of the user profile under which the job was started. It can come from several different sources depending on the type of job. This may be different than the user profile under which the job is currently running. See the Current user profile field for more information.

User return code. The user-defined return code set by ILE high-level language constructs. An example is the program return code in the C language. This field is scoped to the job and represents the most recent return code set by any thread within the job.

Comparing Job Type, Subtype, and Enhanced Job Type with the Work with Active Job Command

The following table compares the job type, job subtype, and enhanced job type fields returned by the QUSRJOBI API to the type field on the Work with Active Job (WRKACTJOB) command.

WRKACTJOB and QUSRJOBI API Comparison			
Job Type Field	**Job Type**	**Job Subtype**	**»Enhanced Job Type**
ASJ (Autostart)	A	blank	0110
BCH (Batch)	B	blank	0210
BCI (Batch immediate)	B	D	0220
EVK (Started by a program start request)	B	E	0310
INT (Interactive)	I	blank	0910, 0920, 0930, or 0940
M36 (AS/400 Advanced 36 machine server)	B	F	blank
MRT (Multiple requester terminal)	B	T	0230
PJ (Prestart job)	B	J	1610, 1620, or 1630
PDJ (Print driver job)	W	P	2310
RDR (Reader)	R	blank	1810
SYS (System)	S or X	blank	1920
SBS (Subsystem monitor)	M	blank	1910
WTR (Writer)	W	blank	2310
blank (Alternative user subtype--not an active job)	B	U	240«

Change Current Job (QWCCCJOB) API

```
Required Parameter Group:

1    Changed job information    Input      Char(*)
2    Error code                 I/O        Char(*)

Default Public Authority: *USE

Threadsafe: No
```

The Change Current Job (QWCCCJOB) API lets you change information for the current job. The user can change the Cancel or the Exit keys.

Required Parameter Group

Changed job information

> INPUT; CHAR(*)

> The information for the job that you want to change. The information must be in the following format:

> *Number of variable length records*

>> BINARY(4)

>> Total number of all of the variable length records.

> *Variable length records*

>> The fields of the job to change and the data used for the change. For the specific format of the variable length record, see Format for Variable Length Record.

Error code

> I/O; CHAR(*)

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

Format for Variable Length Record

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

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Key identifier

4	4	BINARY(4)	Length of key data
8	8	CHAR(*)	Key data

If the length of the data is longer than the key identifier's 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 identifier'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.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Key data. The data used to change a specific field of the job.

Key identifier. The field of the job to change. Only specific fields of the job can be changed. See Key Identifiers for the list of valid keys.

Length of key data. The length of the data used to change a specific field of the job.

Key Identifiers

The following table lists the valid keys for the key identifier area of the variable length record.

Key ID	Type	Field Description
1	CHAR(1)	Exit key
2	CHAR(1)	Cancel key

Key Identifier Descriptions

Exit key. Whether the Exit key is set as pressed for the job. It must have a value of 0 or 1.

0 The Exit key was not pressed.

1 The Exit key was pressed.

Note: The application or command that was called before this API determines how the key is set.

Cancel key. Whether the Cancel key is set as pressed for the job. It must have a value of 0 or 1.

0 The Cancel key was not pressed.

1 The Cancel key was pressed.

Note: The application or command that was called before this API determines how the key is set.

Error Messages

Message ID	Error Message Text
CPF1863 E	Length of value not valid.
CPF1867 E	Value &1 in list not valid.
CPF1868 E	Value &1 for number of records not valid.
CPF2199 E	&2 not valid for field &1.
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

Change Job (QWTCHGJB) API

Required Parameter Group:

1	Qualified job name	Input	Char(26)
2	Internal job identifier	Input	Char(16)
3	Format name	Input	Char(8)
4	Job change information	Input	Char(*)
5	Error code	I/O	Char(*)

≫Optional Parameter Group:

6	Job or Thread identification information	Input	Char(*)
7	Format of job identification information	Input	Char(8)≪

Default Public Authority: *USE

Threadsafe: Conditional; see [Usage Notes](#).

The Change Job (QWTCHGJB) API changes some of the attributes of a job or thread. The attributes changed are determined by the job change information parameter.

The current value of most of the job attributes can be retrieved with the List Job (QUSLJOB) API, the Open List of Jobs (QGYOLJOB) API, the Retrieve Current Attributes (QWCRTVCA) API, the Retrieve Job Information (QUSRJOBI) API or the Retrieve Thread Attributes (QWTRTVTA) API.

If an error occurs, the error code will have a general failure error and the specific error will need to be retrieved out of the job message queue of the job that issued this API call.

The QWTCHGJB API changes a list of attributes similar to the attributes changed with the Change Job (CHGJOB) command.

Authorities and Locks

Job Authority

> The API must be called from within the job that is being changed, or the caller of the API must be running under a user profile that is the same as the job user identity of the job that is being changed. Otherwise, the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

> The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the [Work Management](#) book on the V5R1 Supplemental Manuals Web site.

> Job control (*JOBCTL) special authority is needed for the following attributes to be changed:
> - ❍ Default wait time

- ❍ Purge
- ❍ Run priority
- ❍ Time slice
- ❍ Time slice end pool

Command Authority

*USE authority to the Change Accounting Code (CHGACGCDE) command is needed to change the job accounting code attribute.

»*Object Authority*

If changing the job queue, *READ authority is required for the object, and *EXECUTE authority is required for the library. If changing the output queue, *READ authority is required for the object, and *EXECUTE authority is required for the library. In addition, the caller must be authorized to the output queue currently associated with the job that is being changed. If changing the sort sequence table, *USE authority is required for the object, and *EXECUTE authority is required for the library.

If using the JOBC0300 format to change attributes that are retrieved from the job description, *USE authority is required for the job description and *EXECUTE authority is required for the library. If changing the ASP group information, *USE authority is required for all ASP devices in the ASP group. If changing the current library or the initial library list, *USE authority is required for the libraries.«

Required Parameter Group

Qualified job name

INPUT; CHAR(26)

The name of the job for which the attributes are to be changed. If this value is *INT, the internal job identifier will be used. The qualified job name has three parts:

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

	*	The job or thread in which this program is running. The rest of the qualified job name parameter must be blank. This special value must be used when using the JOBC0200 or JOBC0300 format.
	INT	The internal job identifier locates the job. The user name and job number must be blank. This is only valid for the JOBC0100 format.
	» *THREAD*	The job information is specified in the Job or thread identification information parameter. This special value must be used when using the JOBC0400 format. The user name, job number, and internal job identifier must be blank when using *THREAD. This is valid for the JOBC0400 format only.«

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.

Internal job identifier

INPUT; CHAR(16)

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

Format name

INPUT; CHAR(8)

The format of the list of job »or thread«attributes to be changed. You must use one of the following format names:

JOBC0100   Basic change job list. To be used when changing the attributes of your job or the attributes of another job. This format will not change any specific attributes of a thread. This format will function the same way that the CHGJOB command works.

JOBC0200   Basic change job list for changing the attributes of your own thread. The special value of '*' must be used for the job name. See [Field Descriptions for Valid Keys](#) for the scope of each of the valid key fields. If no scope is mentioned, the attribute is scoped to the job.

Note: The scope of selected attributes may change over time. Currently there are only a few attributes that are scoped to the thread. The attributes that are not scoped to the thread will be updated at the job level. This will affect all threads running under this job. As attributes become scoped to the thread, attributes changed with this format will then be changed for the current thread and will not affect other threads.

JOBC0300   Change select attributes to user profile values. This format is intended to be used after a swap user profile. The special value of '*' must be used for the job name. This format will be functionally similar to the Change Prestart Job (CHGPJ) command. See [Field Descriptions for Valid Keys](#) for the scope of each of the valid key fields. If no scope is mentioned, the attribute is scoped to the job.

Note: The attributes for a thread will be updated at the job level if the specific attributes are not currently defined at the thread level. Selected attributes may be moved to the thread level in a later release.

» *JOBC0400*   Change select attributes of a specific thread. The special value of '*THREAD' must be used for the job name.«

Job change information

INPUT; CHAR(*)

The information for the job that you want to change. The information must be in the following format:

Number of variable length records

BINARY(4). The total number of all the variable length records. If this value is less than 1,

an error message is returned.

Variable length records.

> The attributes of the job to change and the data used for the change. For the specific format of the variable length record, see [Formats for Variable Length Record](#).

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

Job or thread identification information

> INPUT; CHAR(*)

> The information that is used to identify the thread within a job for which specified attributes are to be changed. See [Format of job or thread identification information](#) for details.

Format of job or thread identification information

> INPUT; CHAR(8)

> The format of the job or thread identification information. The possible format names are:

JIDF0100	See [Format of job or thread identification information](#) for details on the job identification information.
JIDF0200	See [Format of job or thread identification information](#) for details on the job identification information.

> **Note:** If the thread handle is available, Format JIDF0200 provides a faster method of accessing a thread that is not the current thread than Format JIDF0100.≪

Formats for Variable Length Record

The following tables define the format for the variable length records.

JOBC0100, JOBC0200 and ≫JOBC0400≪Format: The layout of the JOBC0100, JOBC0200 and ≫JOBC0400≪ format is the same layout as the information returned with the List Job (QUSLJOB) API.

Offset		Type	Field
Dec	Hex		
These fields repeat, in the order listed, for each key requested.		BINARY(4)	Length of attribute information
		BINARY(4)	Key
		CHAR(1)	Type of data
		CHAR(3)	Reserved

BINARY(4)	Length of data
CHAR(*)	Data
CHAR(*)	Reserved

JOBC0300 Format: The JOBC0300 format is for changing attributes to values specified in either the current user profile or the initial user profile.

Offset		Type	Field
Dec	**Hex**		
These fields repeat for each key requested.		BINARY(4)	Key
		CHAR(10)	Data
		CHAR(2)	Reserved

If the length of the data is longer than the key field's 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.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions for JOBC0100, JOBC0200, JOBC0300 and ≫JOBC0400≪ Formats

Data. The data for the attribute that is to be changed. The data for the JOBC0100, JOBC0200, JOBC0300 and ≫JOBC0400≪ formats is defined in the key list section.

Key. The key for the attribute to be changed. See Valid Keys for the list of valid keys.

Length of data. The length of the data for the key field.

Length of attribute information. The total length of input information for this attribute. This value is used to increment to the next attribute in the list.

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

Type of data. The type of input data. This field is provided to maintain the same format layout that is used in the List Job (QUSLJOB) API. This field will not cause any conversion to occur.

B   The input data is in binary format.

C   The input data is in character format.

Valid Keys

The following table contains a list of the valid keys for the JOBC0100 and JOBC0200 formats. See Field Descriptions for Valid Keys for the descriptions of the valid key fields.

Key	Type	Description
0201	CHAR(10)	Break message handling
0302	BINARY(4)	Coded character set ID
0303	CHAR(8)	Country or region ID
0311	CHAR(10)	Character identifier control
0318	CHAR(15)	≫ Client IP address - IPv4 (job)≪
0405	CHAR(4)	Date format
0406	CHAR(1)	Date separator
0408	CHAR(5)	DDM conversation handling
0409	BINARY(4)	Default wait
0410	CHAR(13)	Device recovery action
0413	CHAR(8)	Decimal format
0901	CHAR(10)	Inquiry message reply
1001	CHAR(15)	Job accounting code
1002	CHAR(7)	Job date
1004	CHAR(20)	Job queue name - qualified
1005	CHAR(2)	Job queue priority
1006	CHAR(8)	Job switches
1007	CHAR(10)	Job message queue full action
1201	CHAR(8)	Language ID
1202	CHAR(1)	Logging level
1203	CHAR(10)	Logging of CL programs
1204	BINARY(4)	Logging severity
1205	CHAR(7)	Logging text
1501	CHAR(20)	Output queue name - qualified
1502	CHAR(2)	Output queue priority
1601	CHAR(10)	Print key format
1602	CHAR(30)	Print text
1603	CHAR(10)	Printer device name
1604	CHAR(4)	Purge
1802	BINARY(4)	Run priority (job)
1804	BINARY(4)	Run priority (thread)
1901	CHAR(20)	Sort sequence table - qualified
1902	CHAR(10)	Status message handling
1911	CHAR(30)	Server type
1920	CHAR(10)	Schedule date
1921	CHAR(8)	Schedule time
1922	CHAR(1)	Server mode for Structured Query Language

≫ 1982	CHAR(10)	Spooled file action ≪
2001	CHAR(1)	Time separator
2002	BINARY(4)	Time slice
2003	CHAR(10)	Time-slice end pool

The following table contains a list of the valid keys for the JOBC0300 format. See Field Descriptions for Valid Keys for the descriptions of the valid key fields.

Key	Type	Description
≫0104	CHAR(10)	ASP group information ≪
0302	CHAR(10)	Coded character set ID
0303	CHAR(10)	Country or region ID
0310	CHAR(10)	Current library
0311	CHAR(10)	Character identifier control
0801	CHAR(10)	Home directory
0910	CHAR(10)	Initial library list
1001	CHAR(10)	Job accounting code
1201	CHAR(10)	Language ID
1210	CHAR(10)	Locale
1501	CHAR(10)	Output queue name
1502	CHAR(10)	Output queue priority
1602	CHAR(10)	Print text
1603	CHAR(10)	Printer device name
1901	CHAR(10)	Sort sequence table
1902	CHAR(10)	Status message handling
2701	CHAR(10)	All keys for JOBC0300 format

≫The following table contains a list of the valid keys for the JOBC0400 format. See Field Descriptions for Valid Keys for the descriptions of the valid key fields.

Key	Type	Description
1804	BIN(4)	Run priority (thread) ≪

Field Descriptions for Valid Keys

All keys for JOBC0300 format. All the keys that are valid for the JOBC0300 format will be changed. The list of keys is subject to change at a later time. If other attributes are added to this key in the future, no change will be needed for them to take effect. If this key (2701) is specified, no other keys can be specified. ≫ Even though specifying this key is similar to specifying a list of all the keys that are valid for the JOBC0300 format, the error handling is different. Errors encountered when changing some of the attributes

will not cause the change request to fail, although a diagnostic message may be sent. Errors with ASP group information (key 0104), Current library (key 0310), and Initial library list (key 0910) are considered critical and will stop the change request. In general, errors that would prevent a job from starting will cause the change request to fail. If multiple threads are active, attributes that cannot be changed in a safe manner will be ignored and a diagnostic message will be sent. See the Usage Notes for a list of attributes that can be changed in a safe manner. ≪ The following keys will be changed by this key:

Key ID	Key Name
≫ 0104	ASP group information≪
0302	Coded character set ID
0303	Country or region ID
0310	Current library
0311	Character identifier control
0801	Home directory
0910	Initial library list
1001	Job accounting code
1201	Language ID
1210	Locale
1501	Output queue name
1502	Output queue priority
1602	Print text
1603	Printer device name
1901	Sort sequence table
1902	Status message handling

The attributes that can be set from the locale (coded character set ID and sort sequence table) based on the locale job attributes field (that is retrieved out of the same user profile specified on the locale field) take precedence over the values that are retrieved from the user profile.

This key is valid for the JOBC0300 format only. The possible values are:

*INLUSR   The value for the attributes is retrieved for the user profile under which this thread was initially running.

*CURUSR   The value for the attributes is retrieved for the user profile under which this thread is currently running

≫**ASP group information.** The name of the auxiliary storage pool (ASP) group that is associated with this thread. The ASP group name is the name of the primary ASP device within the ASP group. The libraries in the independent ASPs in the new ASP group plus the libraries in the system ASP (ASP number 1) and basic user ASPs (ASP numbers 2-32) form the library name space and all libraries in the library list of this thread are required to be in the new library name space. Therefore, when the ASP group is updated, the libraries in the system part of the library list, the product libraries, the current library and the libraries in the user part of

the library list will also be updated.

The libraries currently in the system part of the library list that are found in the system ASP or any configured basic user ASP are used as the new system part of the library list. The product libraries that are found in the system ASP or any configured basic user ASP are used as the new product libraries. If the current library is being changed (either with key 0310 or key 2701), the value specified will be used. Otherwise, the library name in the current library entry of the library list is used as the new current library if the library is found in the system ASP or any configured basic user ASP. If the library name in the current entry is not found in the system ASP or any basic user ASP, the current library entry will be removed from the library list. If the user part of the library list is being changed (either with key 0910 or key 2701), the value specified will be used. Otherwise, the libraries currently in the user part of the library list that are found in the system ASP or any configured basic user ASP are used as the new user part of the library list.

This key is valid for the JOBC0300 format only. The following values are possible:

INLUSR   The ASP group information specified in the job description of the user profile under which this thread was initially running is used.

CURUSR   The ASP group information specified in the job description of the current user profile for this thread is used.

Break message handling. How this job handles break messages. This key isvalid for the JOBC0100 and JOBC0200 formats only . The possible values are:

NORMAL   The message queue status determines break message handling.

HOLD   The message queue holds break messages until a user or program requests them. The work station user uses the Display Message (DSPMSG) command to display the messages; a program must issue a Receive Message (RCVMSG) command to receive a message and handle it.

NOTIFY   The system notifies the job's message queue when a message arrives. For interactive jobs, the audible alarm sounds if there is one, and the message-waiting light comes on.

Character identifier control. The character identifier control for the job. This attribute controls the type of CCSID conversion that occurs for display files, printer files, and panel groups. The *CHRIDCTL special value must be specified on the CHRID command parameter on the create, change, or override command for display files, printer files, and panel groups before this attribute will be used.

The possible values for the JOBC0100 and JOBC0200 formats are:

DEVD   The *DEVD special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

JOBCCSID   The *JOBCCSID special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

SYSVAL   The value in the QCHRIDCTL system value will be used.

USRPRF   The CHRIDCTL specified in the user profile under which this thread was initially running will be used.

The possible values for the JOBC0300 format are:

INLUSR   The CHRIDCTL specified in the user profile under which this thread was initially running is used.

CURUSR   The CHRIDCTL specified in the current user profile for this thread is used.

≫Client IP address - IPv4 (job). The IPv4 address of the client for which the thread of this server is servicing currently. This key is valid for the JOBC0200 format only. A value of blanks indicates that the thread is not currently servicing a client. A value of hexadecimal zeros is not allowed. An address is expressed in standard dotted-decimal form www.xxx.yyy.zzz; for example, 130.99.128.1. This field is not required to be an IP address. A change to this attribute in a secondary thread is possible, however, it is essentially meaningless as only the attribute for the inital thread can be retrieved using the Retrieve Job Information (QUSRJOBI) API. For further information on retrieving the *Client IP address - IPv4 or IPv6* that has been implicity set by the operating system, see the [Retrieve Thread Attribute](#) (QWTRTVTA) API. ≪

Coded character set ID. The coded character set identifier used for this job.

The possible values for the JOBC0100 and JOBC0200 formats are:

-1	The CCSID specified in the system value QCCSID is used.
-2	The CCSID specified in the user profile under which this thread was initially running is used.
coded-character-set-identifier	Specify the CCSID.

The possible values for the JOBC0300 format are:

INLUSR	The CCSID specified in the user profile under which this thread was initially running is used.
CURUSR	The CCSID specified in the current user profile for this thread is used.

Country or region ID. The country or region identifier associated with this job.

The possible values for the JOBC0100 and JOBC0200 formats are:

SYSVAL	The system value QCNTRYID is used.
USRPRF	The country or region ID specified in the user profile under which this thread was initially running is used.
country-or-region-ID	Specify the country or region identifier to be used by the job.

The possible values for the JOBC0300 format are:

INLUSR	The country or region ID specified in the user profile under which this thread was initially running is used.
CURUSR	The country or region ID specified in the current user profile for this thread is used.

Current library. The name of the current library that is associated with this thread. This key is valid for the JOBC0300 format only. The following values are possible:

INLUSR	The current library specified in the user profile under which this thread was initially running is used.
CURUSR	The current library specified in the current user profile for this thread is used.

Date format. The format that the date is presented in. This key is only valid for the JOBC0100 and JOBC0200 formats. The following values are possible:

SYS   The system value, QDATFMT, is used.

YMD   The date format used is year, month, and day.

MDY   The date format used is month, day, and year.

DMY   The date format used is day, month, and year.

JUL   The date format used is Julian (year and day).

Date separator. The value used to separate days, months, and years when presenting a date. This key is valid for the JOBC0100 and JOBC0200 formats only. The following values are possible:

S   The system value, QDATSEP is used.

'/'   A slash (/) is used for the date separator.

'-'   A dash (-) is used for the date separator.

'.'   A period (.) is used for the date separator.

' '   A blank is used for the date separator.

','   A comma (,) is used for the date separator.

DDM conversation handling. Specifies whether the connections using distributed data management (DDM) protocols remain active when they are not being used. The connections include APPC conversations, active TCP/IP connections or Opti-Connect connections. The DDM protocols are used in Distributed Relational Database Architecture (DRDA) applications, DDM applications, or DB2 Multisystem applications. This key is only valid for the JOBC0100 and JOBC0200 formats. The following values are possible:

KEEP   The system keeps DDM connections active when there are no users, except for the following:

- The routing step ends on the source system. The routing step ends when the job ends or when the job is rerouted to another routing step.
- The Reclaim Distributed Data Management Conversation (RCLDDMCNV) command or the Reclaim Resources (RCLRSC) command runs.
- A communications failure or an internal failure.
- A DRDA connection to an application server not running on an iSeries server ends.

DROP   The system ends a DDM connection when there are no users. Examples include when an application closes a DDM file, or when a DRDA application runs an SQL DISCONNECT statement.

Decimal format. The type of zero suppression and the decimal point character. This key is only valid for the JOBC0100 and JOBC0200 formats. The following values are possible:

SYSVAL   The value in the system value, QDECFMT, is used as the decimal format for this job.

BLANK   Period for decimal, zero suppression.

J   Comma for decimal, one leading zero.

I   Comma for decimal, zero suppression.

Default wait. The default maximum time (in seconds) that a thread in the job waits for a system instruction, such as a LOCK machine interface (MI) instruction, to acquire a resource. This default wait time is used when a wait time is not otherwise specified for a given situation. Normally, this is the amount of time the user is willing to wait for the system before the request is ended. If the job consists of multiple routing steps, a change to this attribute during a routing step does not apply to subsequent routing steps. The valid range is 1 through 9999999. A value of -1 specifies to change to no maximum wait time. This key is valid for the JOBC0100 and JOBC0200 formats only.

Device recovery action. The action taken for interactive jobs when an I/O error occurs for the job's requesting program device. This key is valid for the JOBC0100 and JOBC0200 formats only. The possible values are:

*SYSVAL   The value in the system value, QDEVRCYACN, is used as the device recovery action for this job.

*MSG   Signals the I/O error message to the application and lets the application program perform error recovery.

*DSCMSG   Disconnects the job when an I/O error occurs. When the job reconnects, the system sends to the application program an error message, that indicates that the job has reconnected and that the work station device has recovered.

*DSCENDRQS   Disconnects the job when an I/O error occurs. When the job reconnects, the system sends the End Request (ENDRQS) command to return control to the previous request level.

*ENDJOB   Ends the job when an I/O error occurs. The system sends to the job's log and to the history log (QHST) a message that indicates that the job ended because of a device error.

*ENDJOBNOLIST   Ends the job when an I/O error occurs. There is no job log produced for the job. The system sends to the QHST log a message that indicates that the job ended because of a device error.

Home directory. The name of the home directory for the integrated file system that is associated with this thread. If the home directory associated with this thread was retrieved from the same user profile that is being specified by either *INLUSR or *CURUSR, then the home directory will not be changed for this thread, even if that user profile's home directory has been changed. If you change the home directory, the job's current working directory will not change. This key is valid for the JOBC0300 format only. The following values are possible:

*INLUSR   The home directory specified in the user profile under which this thread was initially running is used.

*CURUSR   The home directory specified in the current user profile for this thread is used.

Initial library list. The initial user part of the library list that is associated with this thread. This key is only valid for the JOBC0300 format. The possible values are:

*INLUSR   The initial library list specified in the job description of the user profile under which this thread was initially running is used.

*CURUSR   The initial library list specified in the job description of the user profile under which this thread is currently running is used.

Inquiry message reply. How the job answers inquiry messages. This key is only valid for the JOBC0100 and JOBC0200 formats. The possible values are:

RQD   The job requires an answer for any inquiry messages that occur while this job is running.

DFT   The system uses the default message reply to answer any inquiry messages that are issued while this job is running. The default reply is either defined in the message description or is the default system reply.

SYSRPYL   The system reply list is checked to see if there is an entry for an inquiry message that was issued while this job is running. If a match occurs, the system uses the reply value for that entry. If no entry exists for that message, the system uses an inquiry message.

Job accounting code. An identifier assigned to the job by the system to collect resource use information for the job when job accounting is active. The user who is changing this field must have authority to the CHGACGCDE CL command. If the user does not have the proper authority, this field is ignored and processing continues.

The possible values for the JOBC0100 and JOBC0200 formats are:

BLANK   The accounting code is changed to all blanks.

accounting-code Specify the 15-character accounting code used for the next accounting segment. The accounting code may contain alphabetic or numeric characters.

The possible values for the JOBC0300 format are:

INLUSR   The accounting code specified in the job description of the user profile under which this thread was initially running is used.

CURUSR   The accounting code specified in the job description of the user profile under which this thread is currently running is used.

Job date. The date that is assigned to the job. It is in the format CYYMMDD where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*. This value will only be changed for jobs whose status is *JOBQ or *ACTIVE. This key is only valid for the JOBC0100 and JOBC0200 formats.

Job message queue full action. The action to take when the message queue is full. This key is only valid for the JOBC0100 and JOBC0200 formats. The possible values are:

SYSVAL   The value specified for the QJOBMSGQFL system value is used.

NOWRAP   When the job message queue is full, do not wrap. This action causes the job to end.

WRAP   When the job message queue is full, wrap to the beginning and start filling again.

PRTWRAP   When the job message queue is full, wrap the message queue and print the messages that are being overlaid because of the wrapping.

Job queue name - qualified. The qualified name of the job queue that the job is on currently. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. This value is valid for jobs whose status is *JOBQ. For jobs with a status of *OUTQ or *ACTIVE, an error will be signaled. This key is valid for the JOBC0100 and JOBC0200 formats only.

Job queue name CHAR(10). The specific name of the job queue the job is to be on.

Library name	CHAR(10). The name of the library where the job queue is located. This value must be left-justified and padded with blanks. The possible values are:

	*LIBL	All libraries in the job's library list are searched until the first match is found.
	*CURLIB	The current library for the job is used to locate the job queue. If no library is specified as the current library for the job, QGPL is used.
	library-name	Specify the name of the library where the job queue is placed.

Job queue priority. The scheduling priority of the job compared to other jobs on the same job queue. The highest priority is 0 and the lowest is 9. This value is valid for jobs whose status is *JOBQ or *ACTIVE. For jobs with a status of *OUTQ, an error will be signaled. This key is only valid for the JOBC0100 and JOBC0200 formats.

Job switches. The current setting of the job switches that are used by this job. Specify any combination of eight 0's, 1's, or X's to change the job switch settings. If a switch value is not being changed, enter an X in the position that represents that switch. This key is only valid for the JOBC0100 and JOBC0200 formats.

Language ID. The language identifier that is associated with this job. The language identifier is used when *LANGIDUNQ or *LANGIDSHR is specified on the sort sequence parameter. If the job CCSID is 65535, this parameter is also used to determine the value of the job default CCSID.

The possible values for the JOBC0100 and JOBC0200 formats are:

*SYSVAL	The system value QLANGID is used.
*USRPRF	The language ID specified in the user profile under which this thread was initially running is used.
language-ID	Specify the language identifier to be used by the job.

The possible values for the JOBC0300 format are:

*INLUSR	The language ID specified in the user profile under which this thread was initially running is used.
*CURUSR	The language ID specified in the user profile under which this thread is currently running is used.

Locale. The path name of the locale that is assigned to the LANG environment variables. Several job attributes can be set from the locale based on the values from the locale job attributes (locale job attributes are retrieved from the same user profile as the locale). The attributes that can be changed are CCSID, date format, date separator, sort sequence, time separator, and decimal format. This key is only valid for the JOBC0300 format. The possible values are:

*INLUSR	The locale specified in the user profile under which this thread was initially running will be used.
*CURUSR	The locale specified in the user profile under which this thread is currently running will be used.

Logging level. What type of information is logged. This key is valid for the JOBC0100 and JOBC0200 formats only. The possible values are:

0   No messages are logged.

1   All messages sent to the job's external message queue with a severity greater than or equal to the message logging severity are logged. This includes the indications of job start, job end, and job completion status.

2   The following information is logged:

- Logging level 1 information
- Request messages that result in a high-level message with a severity code greater than or equal to the logging severity that caused the request message and all associated messages to be logged.

  Note: A high-level message is one that is sent to the program message queue of the program that receives the request message. For example, QCMD is an IBM-supplied request processing program that receives request messages.

3   The following information is logged:

- Logging level 1 and 2 information
- All request messages
- Commands run by a CL program are logged if it is allowed by the logging of CL programs job attribute and the log attribute of the CL program.

4   The following information is logged:

- All request messages and all messages with a severity greater than or equal to the message logging severity, including trace messages.
- Commands run by a CL program are logged if it is allowed by the logging of CL programs job attribute and the log attribute of the CL program.

Logging of CL programs. Whether or not commands are logged for CL programs that are run. The possible values are *YES and *NO. This key is valid for the JOBC0100 and JOBC0200 formats only.

Logging severity. The severity level that is used in conjunction with the logging level to determine which error messages are logged in the job log. The values range from 00 through 99. This key is valid for the JOBC0100 and JOBC0200 formats only.

Logging text. The level of message text that is written in the job log when a message is logged according to the logging level and logging severity. This key is valid for the JOBC0100 and JOBC0200 formats only. The possible values are:

MSG   Only the message text is written to the job log.

SECLVL Both the message text and the message help (cause and recovery) of the error message are written to the job log.

NOLIST If the job ends normally, no job log is produced. If the job ends abnormally (if the job end code is 20 or higher), a job log is produced. The messages that appear in the job log contain both the message text and the message help.

Output queue name. The name of the default output queue that is used for spooled output produced by this job. The default output queue is only for spooled printer files that specify *JOB for the output queue. The possible values for the JOBC0300 format are:

INLUSR   The output queue specified in the job description of the user profile under which this thread was initially running is used.

CURUSR   The output queue specified in the job description of the user profile under which this thread is currently running is used.

Output queue name - qualified. The qualified name of the default output queue that is used for spooled output produced by this job. The default output queue is only for spooled printer files that specify *JOB for the output queue. The format of the qualified name is a 10-character simple object name followed by a 10-character library name.

Output queue name.

> CHAR(10). The specific name of the output queue that is used. If a special value is specified, it must be the only value in the field.
>
> The possible values for the JOBC0100 and JOBC0200 formats are:
>
> | | |
> |---|---|
> | *DEV* | The DEV parameter is determined by one of these printer file commands: Create Printer File (CRTPRTF), Change Printer File (CHGPRTF), or Override with Printer File (OVRPRTF). |
> | *WRKSTN* | The default output queue that is used with this job is the output queue that is assigned to the work station associated with the job at the time the job is started. |
> | *USRPRF* | The output queue name specified in the user profile under which this thread was initially running is used. |
> | *output-queue-name* | The name and library of the default output queue that is used by the job. Specify the library name last (left-adjusted and padded with blanks) preceded by the output queue name. |

Output queue library name.

> CHAR(10). The name of the library that contains the output queue. The library name must follow the output queue name. The possible values are:
>
> | | |
> |---|---|
> | *LIBL* | All libraries in the job's library list are searched until the first match is found. |
> | *CURLIB* | The current library for the job is used to locate the name of the spooled output queue. If no library is specified as the current library for the job, QGPL is used. |
> | *library-name* | Specify the name of the library where the spooled output queue is located. |

Output queue priority. The output priority for spooled output files that this job produces. The highest priority is 0, and the lowest is 9.

The possible CHAR(2) values for the JOBC0100 and JOBC0200 formats are:

output-priority	A value, ranging from 1 through 9, for the priority of the job's output files. The output priority specified cannot be higher than the priority specified in the user profile under which the job is running.

The possible CHAR(10) values for the JOBC0300 format are:

INLUSR	The output priority specified in the job description of the user profile under which this thread was initially running is used.

CURUSR   The output priority specified in the job description of the user profile under which this thread is currently running is used.

Printer device name. The printer device used for printing output from this job.

The possible values for the JOBC0100 and JOBC0200 formats are:

SYSVAL   The value in the system value QPRTDEV is used as the printer device.

WRKSTN   The default printer device used with this job is the printer device assigned to the work station that is associated with the job.

USRPRF   The printer device name specified in the user profile under which this thread was initially running is used.

printer-device-name   The name of the printer device that is used with this job.

The possible values for the JOBC0300 format are:

INLUSR   The printer device that is specified in the job description of the user profile under which this thread was initially running is used.

CURUSR   The printer device specified in the job description of the user profile under which this thread is currently running is used.

Print key format. Whether border and header information is provided when the Print key is pressed. This key is only valid for the JOBC0100 and JOBC0200 formats. The possible values are:

SYSVAL   The value specified on the system value QPRTKEYFMT determines whether header or border information is printed.

NONE   The border and header information is not included with output from the Print key.

PRTBDR   The border information is included with output from the Print key.

PRTHDR   The header information is included with output from the Print key.

PRTALL   The border and header information is included with output from the Print key.

Print text. The line of text (if any) that is printed at the bottom of each page of printed output for the job. The possible values for the JOBC0100 and JOBC0200 formats are:

SYSVAL   The system value, QPRTTXT, is used.

BLANK   No text is printed on printed output.

print-text   The character string that is printed at the bottom of each page. A maximum of 30 characters can be entered.

The possible values for the JOBC0300 format are:

INLUSR   The print text specified in the job description of the user profile under which this thread was initially running is used.

CURUSR   The print text specified in the job description of the user profile under which this thread is currently running is used.

Purge. Whether or not the job is eligible to be moved out of main storage and put into auxiliary storage at the end of a time slice or when entering a long wait (such as waiting for a work station user's response). This attribute is ignored when more than one thread is active within the job. If the job consists of multiple routing steps, a change to this attribute during a routing step does not apply to subsequent routing steps. This key is valid for the JOBC0100 and JOBC0200 formats only. The possible values are:

YES The job is eligible to be moved out of main storage and put into auxiliary storage. A job with multiple threads, however, is never purged from main storage.

NO The job is not eligible to be moved out of main storage and put into auxiliary storage. When main storage is needed, however, pages belonging to a thread in this job may be moved to auxiliary storage. Then, when a thread in this job runs again, its pages are returned to main storage as they are needed.

Run priority (job). The priority at which the job or thread competes for the processing unit relative to other jobs and threads that are active at the same time. The run priority ranges from 1 (highest priority) to 99 (lowest priority). This value represents the relative (not absolute) importance of the job or thread. For example, a run priority of 25 is not twice as important as a run priority of 50. If the job consists of multiple routing steps, a change to this attribute during a routing step does not apply to subsequent routing steps. This key is valid for the JOBC0100 and JOBC0200 formats only. ≫This key can be used to change the current thread using the JOBC0200 format, but when changing a specific thread, the JOBC0400 format should be used with the *Run priority (thread)* key.≪

The possible values for the JOBC0100 format are:

priority The run priority of the job is changed. The range of values is 1 (highest priority) to 99 (lowest priority). The value may never be higher than the run priority for the job in which the thread is running. If a priority higher than the job's is entered, an error is returned. Changing the run priority of the job affects the run priorities of all threads within the job. For example, the job is running at priority 10, thread A within the job is running at priority 10, and thread B within the job is running at priority 15. The priority of the job is changed to 20. The priority of thread A would then be adjusted to 20 and the priority of thread B would be adjusted to 25.

The possible values for the JOBC0200 format are:

-1 The run priority of the current thread will be set equal to the priority of the job. The thread cannot have a lower priority than its corresponding job.

priority The run priority of the thread is changed. The range of values are the current job's run priority (highest priority) to 99 (lowest priority). If a priority that is higher than the job's is entered, an error is returned.

≫**Run priority (thread).** The run priority for the thread relative to the priority of the other threads that are running in the system. The range of values are from 1 (highest priority) to 99 (lowest priority). The value may never be higher than the run priority for the job in which the thread is running. If a priority higher than the job's is entered, an error is returned. This key is valid for the JOBC0200 and JOBC0400 format only.≪

Schedule date. The date on which the submitted job becomes eligible to run.

If your system or your job is configured to use the Julian date format, *MONTHSTR and *MONTHEND are calculated as if the system or job did not use the Julian date format. This key is only valid for the JOBC0100 format. The possible values are:

CURRENT The submitted job becomes eligible to run on the current date.

MONTHSTR	The submitted job becomes eligible to run on the first day of the month. If you specify *MONTHSTR and if today is the first day of the month and the time you specify on the schedule time parameter has not passed, the job becomes eligible to run today. Otherwise, the job becomes eligible on the first day of the next month.
MONTHEND	The submitted job becomes eligible to run on the last day of the month. If you specify *MONTHEND and if today is the last day of the month and the time you specify on the schedule time parameter has not passed, the job becomes eligible to run today. Otherwise, the job becomes eligible on the last day of the next month.
MON	The job becomes eligible to run on Monday.
TUE	The job becomes eligible to run on Tuesday.
WED	The job becomes eligible to run on Wednesday.
THU	The job becomes eligible to run on Thursday.
FRI	The job becomes eligible to run on Friday.
SAT	The job becomes eligible to run on Saturday.
SUN	The job becomes eligible to run on Sunday.
date	Specify a date in the format CYYMMDD where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*.

Schedule time. The time on the scheduled date at which the job becomes eligible to run. This key is valid for the JOBC0100 format only.

Note: Although the time can be specified to the second, the load on the system may affect the exact time at which the job becomes eligible to run.

The possible values are:

CURRENT	The job is submitted on the current time.
time	The time you want the job to start. The time is specified in 24-hour format as follows:
	Specify a string of 6 digits (HHMMSS) where HH equals hours, MM equals minutes, and SS equals seconds. Valid values for HH range from 00 to 23. Valid values for MM and SS range from 00 to 59.

Server mode for Structured Query Language. Whether or not Structured Query Language (SQL) statements should run in a separate server job. This key is only valid for the JOBC0200 format. The possible values are:

0	The SQL statements will not run in a separate server job.
1	The SQL statements will run in a separate server job. Each SQL connection will be allowed to run with a different user profile and separate transaction scoping.

Server type. The type of server represented by the job. This key is valid for the JOBC0200 format only. A value of blanks indicates that the job is not part of a server. A value of hexadecimal zeros is not allowed. IBM servers start with QIBM.

Sort sequence table. The sort sequence table to be used for string comparisons for this job. The possible values for the JOBC0300 format are:

INLUSR   The sort table specified in the user profile under which this thread was initially running is used.

CURUSR   The sort table specified in the user profile under which this thread is currently running is used.

Sort sequence table - qualified. The qualified name of the sort sequence table to be used for string comparisons for this job. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. The sort sequence table consists of 2 parts:

Sort sequence table name

CHAR(10). The specific name of the sort sequence table. The possible values for the JOBC0100 and JOBC0200 formats are:

SYSVAL   The system value QSRTSEQ is used.

USRPRF   The sort sequence table specified in the user profile under which this thread was initially running is used.

HEX   A sort sequence table is not used. The hexadecimal values of the characters are used to determine the sort sequence.

LANGIDUNQ   A unique-weight sort table is used.

LANGIDSHR   A shared-weight sort table is used.

table-name   The name of the sort sequence table to be used with this job. The table name must be preceded by the library name, left-adjusted, and padded with blanks.

Sort sequence library

CHAR(10). The sort sequence table library that is associated with this job. The possible values are:

LIBL   All libraries in the job's library list are searched until the first match is found.

CURLIB   The current library for the job is searched. If no library is specified as the current library for the job, the QGPL library is used.

library-name The name of the library to be searched. This must be specified after the sort sequence table name and must be left-adjusted and padded with blanks.

»**Spooled file action.** Whether spooled files can be accessed through job interfaces once a job has completed its normal activity.

KEEP   Spooled files are kept with the job when the job completes its activity.

DETACH   Spooled files are detached from the job when the job completes its activity.

SYSVAL   The job will take the spooled file action specified by the QSPLFACN system value.«

Status message handling. Whether you want status messages displayed for this job. The possible values for the JOBC0100 and JOBC0200 formats are:

SYSVAL   The system value QSTSMSG is used.

USRPRF   The status message handling that is specified in the user profile under which this thread was initially running is used.

NONE   This job does not display status messages.

NORMAL   This job displays status messages.

The possible values for the JOBC0300 format are:

INLUSR   Status messages are shown or not shown as specified in the user profile under which this thread was initial running.

CURUSR   Status messages are shown or not shown as specified in the current user profile under which this thread is running.

Time separator. The value used to separate hours, minutes, and seconds when presenting a time. This key is only valid for the JOBC0100 and JOBC0200 formats. The possible values are:

S   The time separator specified in the system value QTIMSEP is used.

':'   A colon (:) is used for the time separator.

'.'   A period (.) is used for the time separator.

' '   A blank is used for the time separator.

','   A comma (,) is used for the time separator.

Time slice. The maximum amount of processor time (in milliseconds) given to each thread in this job before other threads (in this job or in other jobs) are given the opportunity to run. The time slice establishes the amount of time that is needed by a thread in the job to accomplish a meaningful amount of processing. At the end of the time slice, the thread might be put in an inactive state so that other threads can become active in the storage pool. If the job consists of multiple routing steps, a change to this attribute during a routing step does not apply to subsequent routing steps. Valid values range from 1 through 9999999 (that is, 9 999 999 milliseconds or 9999.999 seconds). Although you can specify a value of less than 8, the system takes a minimum of 8 milliseconds to run a process. If you display a job's run attributes, the time slice value is never less than 8. This key is valid for the JOBC0100 and JOBC0200 formats only.

Time-slice end pool. Whether you want interactive jobs moved to another main storage pool at the end of the time slice. This key is only valid for the JOBC0100 and JOBC0200 formats. The possible values are:

SYSVAL   The value in the system value, QTSEPOOL, is used.

NONE   The job does not move to another main storage pool when it reaches the end of the time slice.

BASE   The job moves to the base pool when it reaches the end of the time slice.

»Format of job or thread identification information

Format JIDF0100 is the format of the information needed to identify the job and the thread for which the thread's attributes will be changed. This format supports several special values that can help in identifying the thread.

Format JIDF0200 is the format of the information needed to identify the thread for which the thread's attributes will be changed. This format is to be used when referencing a specific thread for which you already have the thread handle.

Note: If the thread handle is available, Format JIDF0200 provides a faster method of accessing a thread that is not the current thread than Format JIDF0100.

JIDF0100 Format.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

* The job that this program is running in. The job number and user name must contain blanks.

INT The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread identifier. A value that uniquely identifies a thread within a job. If a thread identifier is specified, a thread indicator must also be specified. If the thread indicator is not 0, this field must contain hexadecimal zeros.

Thread indicator. A value that is used to specify the thread within the job. If a thread indicator is specified, a thread identifier must be specified also. The following values are supported:

0   Information from the thread identifier field should be used.

1   The thread that this program is running in currently should be used.

2   The initial thread of the identified job should be used.

Note: For all of the supported values, the combination of the internal job identifier, job name, job number, and user name fields must also identify the job containing the thread.

User name. A specific user profile name, or blanks when the job name specified is a special value.

JIDF0200 Format.

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4), UNSIGNED	Thread handle
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

*   The job that this program is running in. The job number and user name must contain blanks.

*INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread handle. A value that addresses a particular thread within a job. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread. A valid thread handle must be specified. The thread handle is returned on several other interfaces.

Thread identifier. A value which uniquely identifies a thread within a job. A valid thread identifier must be specified.

User name. A specific user profile name, or blanks when the job name specified is a special value. ≪

Usage Notes

How to Determine the Format to Use

The JOBC0100 format is to be used to change an attribute that is scoped to the job. This format will change the attribute for either the job that the request is issued from or for any other job that is on the system.

The JOBC0200 format is to be used to change an attribute for the thread that the request is being issued from. This will change the attribute at the thread level for attributes that are scoped to the thread and will change the attribute at the job level for attributes that are scoped to the job. For attributes that are scoped to the thread, there may be multiple threads active when the change is requested. The change will only affect the current thread. The other active threads will not be affected by the change. The attributes that are scoped to the job may only be changed if there are no secondary threads active, so as to not affect other threads. If a job attribute needs to be changed by a secondary thread or while secondary threads are active, the JOBC0100 format should be used.

The JOBC0300 format may be used after a set user profile has been done with the Set Profile (QWTSETP) API. This format will perform job-initialization type activities. When a job is started, information for various attributes is retrieved from the user profile that the job is starting under. This format will perform a similar function for either the user profile that the thread is currently running under or for the user profile that the thread was initiated under. For attributes that are scoped to the thread, if this format is called with multiple threads active, the change will only affect the current thread. Attributes that are scoped to the job may only be changed if there are no secondary threads active.

»The JOBC0400 format is to be used to change an attribute that is scoped to the thread. It allows changing a thread other than the current thread as well as the current thread.«

Considerations for Attribute Scope and Thread Safety

In the Attribute Scope and Thread Safety table, the *Attribute* column shows the key identifier and the text description for the attribute.

The *Scope* column shows whether the attribute is scoped to the job or to the thread. Attributes changed with this API may be scoped to the job or to the current thread. Some attributes that are scoped to the job could be moved to the thread level in a future release. If that were to occur, this API would be updated to change the thread attribute.

The *Format* columns indicate whether the attributes are considered to be threadsafe when being changed for that format.

The following describes the terminology used in the *Format* columns:

Threadsafe   In general threads terminology, indicates that an interface may be called safely from either an initial thread or a secondary thread. For this particular API, *threadsafe* indicates that an attribute can always be changed and can be considered correct.

The API may be called from the initial or secondary thread to change the attributes of the current job or a different job. The job whose attributes are being changed may be either single threaded or multithreaded.

Note: When attributes that are marked *threadsafe* and are scoped to the job are changed, the change will affect all threads that are running under that job.

Single threaded only   In general threads terminology, indicates that an interface may be called safely only while the job is running single threaded (that is, no secondary threads are active). For this particular API, an attribute marked as *single threaded only* indicates that the attribute can only be changed by that format when changing one's own attribute and there are no other threads active. The change will not be allowed if the target job is multithreaded.

No   The attribute may not be changed safely. The change will not be allowed if multiple threads are active in the job calling this API or in the target job.

Blank   The attribute is not supported for this request.

Attribute Scope and Thread Safety					
Attribute	**Scope**	**JOBC0100**	**JOBC0200**	**JOBC0300**	**»JOBC0400«**
»0104: ASP group information	Current thread			Threadsafe«	
0201: Break message handling	Job	Threadsafe	Single threaded only		
0302: Coded character set ID	Job	Threadsafe	Single threaded only	Single threaded only	
0303: Country or region ID	Job	Threadsafe	Single threaded only	Single threaded only	
0310: Current library	Current thread			Threadsafe	
0311: Character identifier control	Job	Threadsafe	Single threaded only	Single threaded only	
» 0318: Client IP address - IPv4 (job)	Job		Initial thread only[1] «		
0405: Date format	Job	Threadsafe	Single threaded only		
0406: Date separator	Job	Threadsafe	Single threaded only		
0408: DDM conversation	Job	No	No		
0409: Default wait	Job	Threadsafe	Single threaded only		
0410: Device recovery action	Job	Threadsafe	Single threaded only		

0413: Decimal format	Job	Threadsafe	Single threaded only		
0801: Home directory	Current thread			Threadsafe	
0901: Inquiry message reply	Job	Threadsafe	Single threaded only		
0910: Initial library list	Current thread			Threadsafe	
1001: Job accounting code	Job	No	No	No	
1002: Job date	Job	Threadsafe	Single threaded only		
1004: Job queue name - qualified	Job	Threadsafe	Single threaded only		
1005: Job queue priority	Job	Threadsafe	Single threaded only		
1006: Job switches	Job	Threadsafe	Single threaded only		
1007: Job message queue full action	Job	Threadsafe	Single threaded only		
1201: Language ID	Job	Threadsafe	Single threaded only	Single threaded only	
1202: Logging level	Job	Threadsafe	Single threaded only		
1203: Logging of CL programs	Job	Threadsafe	Single threaded only		
1204: Logging severity	Job	Threadsafe	Single threaded only		
1205: Logging text	Job	Threadsafe	Single threaded only		
1210: Locale	Job			No	
1501: Output queue name	Job			Single threaded only	
1501: Output queue name - qualified	Job	Threadsafe	Single threaded only		
1502: Output queue priority	Job	Threadsafe	Single threaded only	Single threaded only	
1601: Print key format	Job	Threadsafe	Single threaded only		
1602: Print text	Job	Threadsafe	Single threaded only	Single threaded only	
1603: Printer device name	Job	Threadsafe	Single threaded only	Single threaded only	
1604: Purge	Job	Threadsafe	Single threaded only		
1802: Run priority (job)	Job	Threadsafe			
1802: Run priority (job)	Current thread		Threadsafe		
》 1804: Run priority (thread)	Thread				Threadsafe《

1901: Sort sequence table	Job			Single threaded only	
1901: Sort sequence table - qualified	Job	Threadsafe	Single threaded only		
1902: Status message handling	Job	Threadsafe	Single threaded only	Single threaded only	
1911: Server type	Job		Single threaded only		
1920: Schedule date	Job	No			
1921: Schedule time	Job	No			
1922: Server mode for Structured Query Language	Job		Single threaded only		
≫ 1982: Spooled file action	Job	Threadsafe	Single threaded only ≪		
2001: Time separator	Job	Threadsafe	Single threaded only		
2002: Time slice	Job	Threadsafe	Single threaded only		
2003: Time-slice end pool	Job	Threadsafe	Single threaded only		
2701: All keys for JOBC0300 format	See the specific keys in this table			See the specific keys in this table	

≫ [1] A change to this attribute in a secondary thread is possible; however, it is essentially meaningless as only the attribute for the inital thread can be retrieved using the Retrieve Job Information (QUSRJOBI) API. This key has no correlation to the attribute set by the system. For further information on retrieving the *Client IP address - IPv4 or IPv6* that has been implicity set by the operating system, see the Retrieve Thread Attribute (QWTRTVTA) API. ≪

Error Messages

Message ID	Error Message Text
CPD0912 D	Printer device &1 not found.
CPD1102 D	Change to &1 only allowed for interactive jobs.
CPD1104 D	Changing &1 to *WRKSTN only allowed for interactive jobs.
CPD1612 D	Not able to allocate job description &1 in &2.
CPF1075 D	Job description &1 in &2 is not found.
CPF1077 D	Not authorized to job description &1 in library &2.
CPF1134 D	Job &3/&2/&1 priority &4 exceeds priority limit &5.

CPF1135 D	Job &3/&2/&1 output priority &4 exceeds priority limit &5.
CPF1144 D	Job queue &1 in library &2 not found.
CPF1145 D	Job queue &1 in library &2 not accessible.
CPF1146 D	User &1 not authorized to job queue &2 in library &3.
CPF1156 D	Job &3/&2/&1 job switch &4 not valid.
CPF1160 D	Job priority not changed.
CPF1252 D	Output queue &1 in library &2 not found.
CPF1253 D	Output queue &1 in library &2 not accessible.
CPF1254 D	User &1 not authorized to output queue &2 in library &3.
CPF1255 D	Output queue library &2 not found.
CPF1264 E	User profile for user name &1 not accessible.
CPF1317 E	No response from subsystem for job &3/&2/&1.
CPF1321 E	Job &1 user &2 job number &3 not found.
CPF1334 D	BRKMSG(*NOTIFY) only valid for interactive jobs.
CPF1335 D	Job queue not changed. Job &3/&2/&1 not batch job.
CPF1337 E	&3/&2/&1 not authorized to change parameters.
CPF1339 D	Job queue not changed. Job &3/&2/&1 not on job queue.
CPF1340 E	Job control function not performed.
CPF1343 E	Job &3/&2/&1 not valid job type for function.
CPF1344 E	Not authorized to control job &3/&2/&1.
CPF1351 E	Function check occurred in subsystem for job &3/&2/&1.
CPF1352 E	Function not done. &3/&2/&1 in transition condition.
CPF1618 E	Job description &1 in library &2 damaged.
CPF1635 D	Requested change no longer allowed.
CPF1644 D	Scheduled date and time not changed.
CPF1650 D	Both scheduled date and time must be changed.
CPF1651 E	Sort sequence table not accessed.
CPF180C E	Function &1 not allowed.
CPF1846 D	CHGJOB did not complete. System value not available.
CPF1854 E	Value &1 for CCSID not valid.
CPF188F E	Not authorized to change job accounting code.

CPF1893 E	Errors occurred while changing job &3/&2/&1.
CPF1895 D	Incorrect format specified with the internal job identifier.
CPF1896 D	Incorrect job name specified.
CPF1897 D	Data for key field &1 not valid.
CPF1898 D	Key field &1 not valid.
CPF1899 D	No other key allowed when specifying key &1.
CPF189A D	Reserved field must be blanks.
CPF189B D	Length field not valid.
CPF189C D	&1 not valid for the data type field.
CPF189E D	Key field &1 not valid with format &2.
CPF189F D	Request not completed.
» CPF18BF E	Thread & not found. «
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
» CPF3C36 E	Number of parameters, &1, entered for this API was not valid. «
» CPF3C3B E	Value for parameter &2 for API &1 not valid. «
CPF3C51 E	Internal job identifier not valid.
CPF3C52 E	Internal job identifier no longer valid.
CPF3C58 E	Job name specified is not valid.
CPF3C59 E	Internal identifier is not blanks and job name is not *INT.
CPF3C88 E	Number of variable length records &1 is 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.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.

CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
» CPFB8E9 E	ASP group &1 not set for thread &2. «

API introduced: V4R2

Change Pool Attributes (QUSCHGPA) API

Required Parameter Group:

1	System pool identifier	Input	Binary(4)
2	New pool size	Input	Binary(4)
3	New pool activity level	Input	Binary(4)

Optional Parameter Group 1:

4	Message logging	Input	Char(1)
5	Error code	I/O	Char(*)

Optional Parameter Group 2:

6	Paging option	Input	Char(10)

Optional Parameter Group 3:
Note: Group 3 is valid for shared pools only.

7	Priority	Input	Binary(4)
8	Minimum pool size %	Input	Binary(4)
9	Maximum pool size %	Input	Binary(4)
10	Minimum faults	Input	Binary(4)
11	Per-thread faults	Input	Binary(4)
12	Maximum faults	Input	Binary(4)

Default Public Authority: *USE

Threadsafe: No

The Change Pool Attributes (QUSCHGPA) API changes the size, activity level, and paging options of any system storage pool. In addition, QUSCHGPA changes the tuning parameters for system storage pools that are also shared pools. A system storage pool identifier is returned with the Materialize Resource Management Data (MATRMD) machine interface (MI) or Retrieve System Status (QWCRSSTS) API. (Note that *system* pool identifiers differ from *subsystem* pool identifiers.) Depending on whether the base pool, shared pool, or private subsystem pool is to be changed, the QUSCHGPA API determines the appropriate command to use and then issues that command. This is similar to the function provided on the System Status display, where you can change the system storage pool size and paging options interactively.

You can use the QUSCHGPA API to tune storage pools without having to know which subsystem monitor allocated the pool. In addition, you do not have to determine whether or not a pool is a shared storage pool, unless parameter group 3 is specified. The Work with System Status (WRKSYSSTS), the Work with Subsystems (WRKSBS), and the Work with Shared Pools (WRKSHRPOOL) commands provide similar functions.

Authorities and Locks

Subsystem Description Authority

> *OBJOPR, *OBJMGT, and *READ

Subsystem Description Library Authority

> *EXECUTE

Required Parameter Group

System pool identifier

> INPUT; BINARY(4)

> This identifies which pool is to be changed. This number corresponds to the number returned on option nine of the MATRMD MI instruction using the ILE C language. This also corresponds to the identifier shown on the Work with System Status display. This parameter is a value ranging from 1 through 64, where pool 1 is the machine pool, and pool 2 is the base pool.

New pool size

> INPUT; BINARY(4)

> The size of the pool in kilobytes, where one kilobyte is 1024 bytes. If you do not want the pool size to be changed, you must specify a value of -1 for this parameter. The minimum value is 256 kilobytes.

> **Note:** For compatibility with previous releases, a pool size of 32 through 255 kilobytes can be specified. However, since the minimum pool size is 256 kilobytes, the pool will not be changed when a size of 32 through 255 kilobytes is specified.

New pool activity level

> INPUT; BINARY(4)

> The activity level for the pool. If you do not want the activity level to be changed, you must specify a value of -1 for this parameter. You cannot change the activity level of the machine pool.

Optional Parameter Group 1

Message logging

> INPUT; CHAR(1)

> Whether messages reporting that a change was made are written to the current job's job log and to the QHST message log. This affects the logging of change-related messages only; it does not affect the logging of error messages. Valid values are:

> *Y*   Log change messages.

> *N*   Do not log change messages.

> If this parameter is omitted, Y is used and change messages are logged.

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.

Optional Parameter Group 2

Paging option

> INPUT; CHAR(10)
>
> Whether the system should dynamically adjust the paging characteristics of the storage pool for optimum performance. Valid values are:

*SAME	The paging option for the storage pool is not changed.
*FIXED	The system will not dynamically adjust the paging characteristics; system default values are used.
*CALC	The system will dynamically adjust the paging characteristics.

> If this parameter is omitted, the paging option is not changed.

Optional Parameter Group 3

Note: Group 3 is valid for shared pools only.

Priority

> INPUT; BINARY(4)
>
> The priority of this pool relative to the priority of the other storage pools. Valid values are 1 through 14. The priority for the *MACHINE pool must be 1. This value is used by the system if the performance adjustment (QPFRADJ) system value is set to 2 or 3. If this parameter is omitted, the priority value is not changed. If you want the system to calculate the priority, you must specify -2 for this parameter. If you do not want this value to change, you may specify -1 for this parameter.

Minimum pool size %

> INPUT; BINARY(4)
>
> The minimum amount of storage to allocate to this storage pool (as a percentage of total main storage), specified in hundredths. That is, a value of 1234 means 12.34 percent. This value cannot be greater than the maximum pool size % parameter value. This value is used by the system if the QPFRADJ system value is set to 2 or 3. If this parameter is omitted, the minimum size value is not changed. If you want the system to calculate the minimum size, you must specify -2 for this parameter. If you do not want this value to change, you may specify -1 for this parameter.

Maximum pool size %

> INPUT; BINARY(4)
>
> The maximum amount of storage to allocate to this storage pool (as a percentage of total main storage), specified in hundredths. That is, a value of 1234 means 12.34 percent. This value cannot

be less than the minimum pool size % parameter value. This value is used by the system if the QPFRADJ system value is set to 2 or 3. If this parameter is omitted, the maximum size value is not changed. If you want the system to calculate the maximum size, you must specify -2 for this parameter. If you do not want this value to change, you may specify -1 for this parameter.

Minimum faults

INPUT; BINARY(4)

The minimum faults-per-second guideline to use for this storage pool, specified in hundredths. That is, a value of 1234 means 12.34. This value is used by the system if the QPFRADJ system value is set to 2 or 3. If this parameter is omitted, the minimum faults value is not changed. If you want the system to calculate minimum faults, you must specify -2 for this parameter. If you do not want this value to change, you may specify -1 for this parameter.

Per-thread faults

INPUT; BINARY(4)

The faults per second for each active thread in this storage pool, specified in hundredths. That is, a value of 1234 means 12.34. Each job is comprised of one or more threads. The system multiplies this number by the number of active threads that it finds in the pool. This result is added to the minimum faults parameter to calculate the faults-per-second guideline to use for this pool. This value is used by the system if the QPFRADJ system value is set to 2 or 3. If this parameter is omitted, the per-thread faults value is not changed. If you want the system to calculate per-thread faults, you must specify -2 for this parameter. If you do not want this value to change, you may specify -1 for this parameter.

Maximum faults

INPUT; BINARY(4)

The maximum faults-per-second guideline to use for this storage pool, specified in hundredths. That is, a value of 1234 means 12.34. The sum of minimum faults and per-thread faults must be less than the value of the maximum faults parameter. This value is used by the system if the QPFRADJ system value is set to 2 or 3. If this parameter is omitted, the maximum faults value is not changed. If you want the system to calculate maximum faults, you must specify -2 for this parameter. If you do not want this value to change, you may specify -1 for this parameter.

The following table summarizes the values you can specify for the system pool identifier, the new pool size, and the new pool activity level.

System Pool Identifier	New Pool Size	New Pool Activity Level
1 (Machine pool)	-1 or >= 256	-1
2 (Base pool)	-1 or >= 256 [1]	-1 or 1 through 32 767
3 to 64	>= 256 [1]	1 through 32 767

[1] For compatibility with previous releases, a pool size of 32 through 255 kilobytes can be specified. Since the minimum pool size is 256 kilobytes, however, the pool will not be changed when a size of 32 through 255 kilobytes is specified.

For pools 3 through 64, both size and pool activity level must be specified.

In some cases, pool size changes do not take effect immediately. For example, a save or restore operation might be using some of the storage allocated to a pool, or the system might be using some of the storage

allocated to the base pool. The size is changed only when the storage being used is free again.

The base pool holds all unused main storage on the system that is not allocated to other shared or private pools. As subsystems are started and allocate storage for their shared and private storage pools, that storage comes from the base pool. The base pool (pool number 2) size is what is left after pool 1 and pools 3 through 64 are subtracted from the total main storage. The QBASPOOL system value is a minimum size, and it is not the actual size of the base pool. At this minimum size, the system does not allow additional storage requests. For this reason, you must calculate the storage requirements for all pools on the system, including the base pool, and then run this API.

Error Messages

Message ID	Error Message Text
CPF1001 E	Wait time expired for system response.
CPF1076 E	Specified value not allowed for system value &1.
CPF1078 E	System value &1 not changed.
CPF113A E	Sum of MINFAULT and JOBFAULT parameters exceeds MAXFAULT parameter.
CPF113B E	Minimum size percentage exceeds maximum size percentage.
CPF113C E	Parameter not valid for private pool.
CPF113E E	Range of parameter &2 does not include &4.
CPF1165 E	Specified parameter not allowed for *MACHINE pool.
CPF1619 E	Subsystem description &1 in library &2 damaged.
CPF1691 E	Active subsystem description may or may not have changed.
CPF1697 E	Subsystem description &1 not changed.
CPF1879 E	Paging option &1 not valid.
CPF1880 E	Machine pool paging option cannot be changed.
CPF1881 E	Changing private pool paging option not allowed.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CA0 E	System pool &1 does not exist.
CPF3CA1 E	Pool size &1 is not valid.
CPF3CA2 E	Activity level &1 is not valid.
CPF3CA3 E	Pool &1 is not in use.
CPF3CA4 E	Changing machine pool activity level is not allowed.
CPF3CA5 E	Both pool size and activity level are required.
CPF3CA6 E	Message logging value &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

Example: Changing System Storage Pool Attributes

The following is an example of how to change system storage pool attributes using the QUSCHGPA API:

```
Pseudocode
```

```
        .
        .
        .
MATRMD (OPERAND1, OPERAND2);
DO               /* Do loop for each pool in use    */
        .
        .               /* Calculate the desired pool size
                           and activity level         */
        .
END
DO               /* Do loop for each pool in use    */
CALL QUSCHGPA (POOLID, POOLSIZE, POOLACTLVL);
                 /* Change pool attributes          */
END
        .
        .
        .
```

API introduced: V1R3

Change Pool Tuning Information (QWCCHGTN) API

Required Parameter Group:

1	System pool identifier	Input	Binary(4)
2	Change request	Input	Char(*)
3	Length of change request	Input	Binary(4)
4	Format name	Input	Char(8)
5	Error code	I/O	Char(*)

Default Public Authority: *EXCLUDE

Threadsafe: No

The Change Pool Tuning Information (QWCCHGTN) API changes information about tuning being performed by the system for the different storage pools. The Materialize Resource Management Data (MATRMD) machine interface (MI) instruction can be used to retrieve the current setting of the tuning parameters.

Required Parameter Group

System pool identifier

> INPUT; BINARY(4)

> The pool is to be changed. This number corresponds to the number returned on option 9 of the Materialize Resource Management Data (MATRMD) MI instruction. This also corresponds to the identifier shown on the Work with System Status display. This parameter is a value ranging from 2 through 64, where pool 2 is the base pool.

Change request

> INPUT; CHAR(*)

> The variable containing the new tuning information. See TUNI0100 Format for the definition of the fields for this parameter.

Length of change request

> INPUT; BINARY(4)

> The length of the change request list. This area must be as large as the format specified.

Format name

> INPUT; CHAR(8)

> The format of the information to be changed. The valid values are:

>> *TUNI0100*   Tuning information for a storage pool.

Error code

     I/O; CHAR(*)

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

TUNI0100 Format

The following table shows the information that must be specified in the change request parameter when format TUNI0100 is specified. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Type of tuning
1	1	CHAR(1)	Change page handling
2	2	BINARY(4)	Blocking factor for nondatabase objects
6	6	CHAR(1)	Allow exchange operations (for class 1 objects in pool)
7	7	CHAR(1)	Type of transfer from main storage to auxiliary storage (for class 1 objects in pool)
8	8	BINARY(4)	Blocking factor for database (class 1) objects
12	C	CHAR(1)	Allow exchange operations (for class 2 objects in pool)
13	D	CHAR(1)	Type of transfer from main storage to auxiliary storage (for class 2 objects in pool)
14	E	BINARY(4)	Blocking factor for database (class 2) objects
18	12	CHAR(1)	Allow exchange operations (for class 3 objects in pool)
19	13	CHAR(1)	Type of transfer from main storage to auxiliary storage (for class 3 objects in pool)
20	14	BINARY(4)	Blocking factor for database (class 3) objects
24	18	CHAR(1)	Allow exchange operations (for class 4 objects in pool)
25	19	CHAR(1)	Type of transfer from main storage (for class 4 objects in pool) to auxiliary storage
26	1A	BINARY(4)	Blocking factor for database (class 4) objects

When tuning is requested (values 1, 2, or 3 for the type of tuning field), the system periodically categorizes database objects into four different performance classes. The classes are:

Class 1   Object access appears to be random. A disk access is required for nearly each record that is accessed.

Class 2   Locality of reference detected. Several records are being accessed per disk access.

Class 3   High locality of reference detected. The object is being processed in a sequential manner; references are highly clustered and large portions of the object are resident in main storage.

Class 4 The class of a database object is adjusted if the object's size is small in comparison to the available storage in the storage pool. This class adjustment involves adding 1 to the class number; therefore, a class 3 database object (as defined above) would be treated as a class 4 if it were small in comparison to the available storage in the storage pool.

Reference information for determining an object's class is collected periodically. It is collected by storage pool because an object's class varies over time and by storage pool.

Note: When a new system pool is created as a result of starting a subsystem, the type of tuning and change page handling attributes for the new system pool are initialized based on the type of storage pool being created. For shared storage pools, the type of tuning and change page handling attributes are set based on the paging option defined for the shared storage pool. For private storage pools, the type of tuning attribute is set to indicate no tuning should be done and the change page handling attribute is set to the system default value.

Field Descriptions

Allow exchange operations. The exchange operation used to reduce the working set size. This is done by overlaying data that is already in main storage with new data this is being brought into main storage. The values for this field are:

0 Use the system default, which is 1 (allow exchange operations)

1 Allow exchange operations

2 Disable exchange operations

3 Disable exchange operations (The data that already exists in main storage should be a good candidate to be replaced when additional storage is needed in the storage pool.)

The value specified for this field is ignored unless static tuning is specified for the type of tuning field.

Blocking factor for database objects. The amount of data that should be brought into main storage when a request is made to read database objects from auxiliary storage. The values for this field are:

0 Use the system default, which is 4 (transfer data into main storage in 4KB blocks)

4 Transfer data into main storage in 4KB blocks

8 Transfer data into main storage in 8KB blocks

16 Transfer data into main storage in 16KB blocks

32 Transfer data into main storage in 32KB blocks

64 Transfer data into main storage in 64KB blocks

128 Transfer data into main storage in 128KB blocks

The system may need to issue multiple I/O operations to bring the data into main storage. The value specified for the blocking factor for database objects field is ignored unless static tuning is specified for the type of tuning field.

Blocking factor for nondatabase objects. The amount of data that should be brought into main storage when a request is made to read nondatabase objects from auxiliary storage. The possible values for this

field are:

0   Use the system default, which is 4 (transfer data into main storage in 4KB blocks)

4   Transfer data into main storage in 4KB blocks

8   Transfer data into main storage in 8KB blocks

16   Transfer data into main storage in 16KB blocks

32   Transfer data into main storage in 32KB blocks

The system may need to issue multiple I/O operations to bring the data into main storage. The value specified for the blocking factor for nondatabase objects is ignored unless static tuning is specified for the type of tuning field.

Change page handling. The method the system uses to determine when to write changed pages to auxiliary storage. The values for this field are:

0 Use the system default, which is 1 (Changed pages should be written to auxiliary storage when there is a demand for pages in a storage pool.)

1 Changed pages should be written to auxiliary storage when there is a demand for pages in a storage pool

2 In addition to writing changed pages on demand, periodically write changed pages to auxiliary storage

Type of transfer from main storage to auxiliary storage. The method the system uses to process a request to write an object to auxiliary storage. The values for this field are:

0 Use the system default, which is 1 (When objects are changed, write the changes to auxiliary storage. Indicate that the portion of the object that was written to auxiliary storage should be a good candidate to be replaced when additional storage is needed in the storage pool.)

1 When objects are changed, write the changes to auxiliary storage. Indicate that the portion of the object that was written to auxiliary storage should be a good candidate to be replaced when additional storage is needed in the storage pool.

2 When objects are changed, write the changes to auxiliary storage.

3 Do not immediately write the changes to auxiliary storage. Indicate that the portion of the object that was changed should be a good candidate to be replaced when additional storage is needed in the storage pool.

4 Do not immediately write the changes to auxiliary storage.

The value specified for this field is ignored unless static tuning is specified for the type of tuning field.

Type of tuning. The method used by the system to tune the storage pool. The values for this field are:

0 No tuning is being performed for this pool.

    All values specified for the blocking factor, the allow exchange operations, and the type of transfer from main storage to auxiliary storage fields are ignored. The system default values are used for all these fields.

1   Static tuning is being performed for this pool. Static tuning implies that the values specified for blocking factor, exchange operation, and transfers to auxiliary storage are not dynamically adjusted by the system.

    Values must be specified for the blocking factor, allow exchange operations, and type of transfer from main storage to auxiliary storage for the storage pools.

2   Dynamic tuning of transfers into main storage is being performed. This indicates that the system is dynamically adjusting the blocking factor and exchange operations.

    Because the values for blocking factor and allow exchange operations are dynamically adjusted, the values specified on the API are ignored. The value used for the transfer to auxiliary storage field is set to ensure that requests to write data to auxiliary storage are processed immediately.

3   Dynamic tuning of transfers into main storage and to auxiliary storage is being performed. This indicates that the system is dynamically adjusting the blocking factor, exchange operations, and transfers to auxiliary storage.

    Because the values for the blocking factor, allow exchange operations, and transfers to auxiliary storage are dynamically adjusted, the values specified on the API are ignored.

Error Messages

Message ID	Error Message Text
CPF1001 E	Wait time expired for system response.
CPF1870 E	Value &1 for type of tuning not valid.
CPF1871 E	Value &1 for change page handling not valid.
CPF1872 E	Value &1 for blocking factor not valid.
CPF1873 E	Value &1 for exchange operation not valid.
CPF1874 E	Value &1 for transfer to auxiliary storage not valid.
CPF1875 E	Value &1 for change request length not valid.
CPF1876 E	Value &1 for pool number not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Change Subsystem Entry (QWDCSBSE) API

<div style="border:1px solid #999; padding:10px;">

Required Parameter Group:

1	Qualified subsystem name	Input	Char(20)
2	Change format name	Input	Char(8)
3	Subsystem entry identifier	Input	Char(*)
4	Change information	Input	Char(*)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

</div>

The Change Subsystem Entry (QWDCSBSE) API changes a subsystem entry in the specified subsystem description.

Authorities and Locks

Job Description Authority
>   *USE

Job Description Library Authority
>   *EXECUTE

Subsystem Description Authority
>   *OBJMGT, *USE

Subsystem Description Library Authority
>   *EXECUTE

User Profile Authority
>   *USE

Required Parameter Group

Qualified subsystem name

>   INPUT; CHAR(20)

>   The subsystem description that contains the subsystem entry being changed. The first 10 characters contain the subsystem description name, and the second 10 characters contain the library name. You can use these special values for the library name:

>   *CURLIB*   The job's current library

>   *LIBL*   The job's library list

Change format name

INPUT; CHAR(8)

The format of the subsystem entry to change. You can use the following format:

SBSE0500   Prestart job entry. For details, see SBSE0500 Format (Prestart Job Entry).

Subsystem entry identifier

INPUT; CHAR(*)

The subsystem entry that is to be changed. The identifier is specific to the entry type. For prestart job entries, see SBSE0500 Format (Prestart Job Entry) for details.

Change information

INPUT; CHAR(*)

The information for the subsystem entry that you want to change. The information must be in the following format:

Number of variable length records

BINARY(4)
The total number of all of the variable length records.

Variable length records

The attributes of the subsystem entry that are to be changed. Refer to Format for Variable Length Record for more information.

Error code

I/O; CHAR(*)

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

Format for Variable Length Record

The following table shows the layout of the variable length record. For a detailed description of each field, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Length of variable length record
4	4	BINARY(4)	Attribute key
8	8	BINARY(4)	Length of data
12	C	CHAR(*)	Data

If the length of the data is longer than the key field's data length, the data is truncated at the right. No message is issued.

If the length of the data is shorter than the key field's data length and the key contains binary data, an error message is issued. If the key does not contain binary data, the field is padded with blanks.

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.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Attribute key. The attribute to be set. For prestart job entries, see SBSE0500 Format (Prestart Job Entry) for details.

Data. The value to which a specific attribute is to be set.

Length of data. The length of the attribute value.

Length of variable length record. The length of the record including this field.

SBSE0500 Format (Prestart Job Entry)

This format changes a prestart job entry in the specified subsystem description. The associated subsystem may be active when the prestart job entry is changed. Changes made to the entry when the subsystem is active are reflected over time. Prestart jobs that are created after the API is issued use the new job-related values.

Subsystem Entry Identifier for SBSE0500 Format

Qualified program name

> CHAR(20)
>
> The qualified name of the program that identifies the prestart job entry being changed. The first 10 characters contain the program name, and the second 10 characters contain the library name. You can use these special values for the library name:
>
> *CURLIB   The job's current library
>
> *LIBL   The job's library list

Attribute Keys for SBSE0500 Format

The following table shows the valid attribute keys for the attribute key field of the variable length record. For a detailed description of each field, see Field Descriptions of Attribute Keys for SBSE0500 Format.

Key	Type	Field
1	CHAR(10)	User profile name
2	CHAR(1)	Start jobs

3	BINARY(4)	Initial number of jobs
4	BINARY(4)	Threshold
5	BINARY(4)	Additional number of jobs
6	BINARY(4)	Maximum number of jobs
7	CHAR(10)	Job name
8	CHAR(20)	Job description name
9	BINARY(4)	Maximum number of uses
10	CHAR(1)	Wait for job
11	BINARY(4)	Pool identifier
12	CHAR(20)	Class 1 name
13	BINARY(4)	Class 1 number of jobs
14	CHAR(20)	Class 2 name
15	BINARY(4)	Class 2 number of jobs

Field Descriptions of Attribute Keys for SBSE0500 Format

Additional number of jobs. The additional number of prestart jobs that are started when the number of prestart jobs drops below the threshold value. The value of this parameter must be less than the value of the maximum number of jobs. Valid values range from 0-999.

Class 1 name. The name of a class under which the prestart jobs run. Two classes can be specified for a prestart job entry, class 1 name and class 2 name. Each class defines the number of jobs that run under that class. See class 1 number of jobs and class 2 number of jobs.

Jobs start under the first class specified until the number of jobs specified for the first class is reached. After the allowed number of jobs specified for the first class is reached, jobs are started under the second class.

The possible values are:

SBSD   The class that has the same name as the subsystem description specified in the qualified subsystem name is used for prestart jobs.

Qualified class name The name of the class used for prestart jobs. The first 10 characters contain the class name, and the second 10 characters contain the library name. You can use these special values for the library name:

   CURLIB   The job's current library

   LIBL   The job's library list

If the class does not exist when the entry is added, a library qualifier must be specified because the qualified class name is retained in the subsystem description.

Class 1 number of jobs. The maximum number of jobs to run that use the first class. If you specified the maximum number of jobs key to be changed, the value for the number of jobs specified for this key might need to be changed. If -3 or -4 is specified, the system recalculates the value for the number of jobs to use the specified class. The possible values are:

-3	*CALC: The system calculates how many prestart jobs use this class. If only one class is specified and -3 is specified, all of the jobs use that class. If two classes are specified and -3 is specified for both, the first class is the value of the maximum number of jobs divided by two, and the second class is the value of the maximum number of jobs minus the value calculated for the first class. If a specific number of jobs is specified for either class and -3 is specified for the other class, the system calculates the difference between maximum number of jobs and the specific number of jobs for the -3 designation.
-4	*MAXJOBS: All prestart jobs use the specified class.
number of jobs	The number of jobs that use this class. The sum of the values specified for class 1 and class 2 number of jobs must equal the value of the maximum number of jobs. If you specify one of the class number of job keys, you may also need to specify the maximum number of jobs keys.

Class 2 name. The name of a class under which the prestart jobs run. Two classes can be specified for a prestart job entry, class 1 name and class 2 name. Each class defines the number of jobs that run under that class. See class 1 number of jobs and class 2 number of jobs.

Jobs start under the first class specified until the number of jobs specified for the first class is reached. After the allowed number of jobs specified for the first class is reached, jobs are started under the second class.

The possible values are:

NONE	This value indicates that only one class is used.
SBSD	The class that has the same name as the subsystem description specified in the qualified subsystem name is used for prestart jobs.
Qualified class name	The name of the class being used for prestart jobs. The first 10 characters contain the class name, and the second 10 characters contain the library name. You can use these special values for the library name:

CURLIB	The job's current library
LIBL	The job's library list

If the class does not exist when the entry is added, a library qualifier must be specified because the qualified class name is retained in the subsystem description.

Class 2 number of jobs. The maximum number of jobs that use the second class. The possible values are:

-3	*CALC: The system calculates how many prestart jobs use this class. If only one class is specified and -3 is specified, all of the jobs use that class. If two classes are specified and -3 is specified for both, the first class is the value of the maximum number of jobs divided by two, and the second class is the value of the maximum number of jobs minus the value calculated for the first class. If a specific number of jobs is specified for either class and -3 is specified for the other class, the system calculates the difference between the maximum number of jobs and the specific number of jobs for the -3 designation.
-4	*MAXJOBS: All prestart jobs use the specified class.

number of jobs The number of jobs that use this class. The sum of the values specified for class 1 and class 2 number of jobs must equal the value of the maximum number of jobs. If you specify one of the class number of job keys, you may also need to specify the maximum number of jobs keys.

Initial number of jobs. The initial number of prestart jobs that are started when the subsystem specified in the qualfified subsystem name is started. The value of this key must be less than or equal to the value of the maximum number of jobs. The value of this key must be greater than or equal to the value of the threshold. Valid values range from 1-9999.

Job description name. The name of the job description being used for the prestart job. If the job description does not exist when the entry is changed, a library qualifier must be specified because the qualified job description name is retained in the subsystem description.

USRPRF	The job description name specified in the user profile for the prestart job entry is used.
SBSD	The job description that has the same name as the subsystem description for this prestart job entry is used.
Qualified job description name	The name of the job description being used for this prestart job. The first 10 characters contain the job description name, and the second 10 characters contain the library name. You can use these special values for the library name:

CURLIB	The job's current library
LIBL	The job's library list

Job name. The name of the prestart job that is started.

PGM	The job name is the same name as the qualified program name specified in the subsystem entry identifier.
job-name	The name of the prestart job.

Maximum number of jobs. The maximum number of prestart jobs that can be active at the same time for this prestart job entry. The value of this key must be greater than or equal to the value of the initial number of jobs. The value of this key must be greater than the value of the additional number of jobs. If the value specified for this key is changed, the value specified for one or both of the class number of job keys might also need to be changed. The possible values follow:

-1	*NOMAX: There is no maximum number of jobs that can be active at the same time.
maximum-jobs	The maximum number of prestart jobs that can be active at the same time. Valid values range from 1-9999.

Maximum number of uses. The maximum number of that can be handled by each prestart job before the subsystem ends the job in a controlled manner. Jobs are ended in a controlled manner by issuing an ENDJOB command with a value of *CNTRLD on the OPTION parameter.

-1	*NOMAX: There is no maximum number of that a prestart job can handle before it is ended. If -1 is specified, the prestart jobs may end abnormally because the job has exceeded the allowed maximum job log size, the maximum number of spooled files, the maximum processor unit time, or the maximum temporary storage space required.

maximum-uses   The maximum number of that a prestart job can handle before it is ended. Valid values range from 1 through 1000.

Pool identifier. The subsystem pool identifier under which the prestart jobs are run. Valid values range from 1 through 10.

Start jobs. Whether prestart jobs are started when the subsystem is started. The possible values are:

0   The prestart jobs are not started at the time the subsystem is started. The Start Prestart Jobs (STRPJ) command must be used to start these prestart jobs.

1   The prestart jobs are started when the subsystem is started.

Threshold. The number at which additional prestart jobs are started. When the pool of available prestart jobs (jobs available to service is reduced below this number, more jobs (specified by the additional number of jobs value) are started and added to the available pool. The value of this key must be less than or equal to the value of the initial number of jobs. Valid values range from 1-9999.

User profile name. The user profile under which the prestart job is initiated. In addition, the current user profile of the prestart job is set to this user whenever the job waits for a request to handle.

Note: When a prestart job is given a request to handle, the current user profile of the job is updated. Refer to the Work Management book on the V5R1 Supplemental Manuals Web site for information on how this profile is determined. This change in current user profile is for authority checking only. None of the other attributes of the user profile, such as the current library (CURLIB) or the initial program to call (INLPGM), are given to the prestart job.

Wait for job. Whether program start requests wait for a prestart job to become available or are rejected if a prestart job is not immediately available when the program start request is received. Refer to the manual for the communications type being used to determine the timing considerations for program start requests. The possible values follow:

0   Program start requests are rejected if a prestart job is not immediately available when the program start request is received.

1   Program start requests wait until a prestart job is available, or a prestart job is started to service the request.

Error Messages

Message ID	Error Message Text
CPF1619 E	Subsystem description &1 in library &2 damaged.
CPF1697 E	Subsystem description &1 not changed.
CPF3C21 E	Format name &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
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.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

»Control Thread (QTHMCTLT) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format of receiver information	Input	Char(8)
4	Job or thread identification information	Input	Char(*)
5	Format of job or thread identification information	Input	Char(8)
6	Action	Input	Binary(4)
7	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Control Thread (QTHMCTLT) API holds, releases, or ends the specified thread.

End thread cannot be specified for the initial thread of a job. Hold thread or release thread cannot be specified for the initial thread of a system job.

Authorities and Locks

Job Authority

If the action to be taken is hold thread or release thread, the caller of the API must be running under a user profile that is the same as the job user identity of the job containing the thread for which the specified action is to be taken. Otherwise, the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority, or be authorized to the Thread Control function of Operating System/400 through iSeries Navigator's Application Administration support.

If the action to be taken is end thread, the caller of the API must be running under a user profile that has service (*SERVICE) special authority or be authorized to the Thread Control function of Operating System/400 through iSeries Navigator's Application Administration support.

The Change Function Usage Information (QSYCHFUI) API, with a function ID of QIBM_SERVICE_THREAD, can be used to change the list of users that are allowed to end, hold, or release a thread.

The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the Work Management book on the V5R1 Supplemental Manuals Web site.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

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

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of receiver information

> INPUT; CHAR(8)

> The format of the information returned in the receiver variable. The format name is:

> *CTLT0100*   See [CTLT0100 Format](#) for details on the information returned.

Job or thread identification information

> INPUT; CHAR(*)

> The information that is used to identify the thread within a job for which an action is to be taken. See [Format of job or thread identification information](#) for details.

Format of job or thread identification information

> INPUT; CHAR(8)

> The format of the job or thread identification information. The possible format names are:

> *JIDF0100*   See [JIDF0100 Format](#) for details on the job identification information.

> *JIDF0200*   See [JIDF0200 Format](#) for details on the job identification information.

> **Note:** If the thread handle is available, Format JIDF0200 provides a faster method of accessing a thread that is not the current thread than Format JIDF0100.

Action

> INPUT; BINARY(4)

> The action to be taken against the thread. The following actions are supported:

> *1*   Hold thread

> *2*   Release thread

> *3*   End thread

> The end thread, hold thread, and release thread actions are asynchronous operations. A portion of

the action is done by the current thread, and the remainder of the action is done by the target thread. When control is returned to the current thread, the action may not have been performed in its entirety. Since some processing must be performed in the target thread, the action could be delayed for some period of time if higher priority threads in this or other jobs prevent the target thread from running.

End thread cannot be specified for the initial thread of a job. Hold thread or release thread cannot be specified for the initial thread of a system job.

Error code

I/O; CHAR(*)

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

CTLT0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	UNSIGNED BINARY(4)	Hold count

Field Descriptions for CTLT0100 Format

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.

Hold count. The number of times the thread has been held prior to performing the action. The hold count is the count of fully processed hold operations currently in effect for the thread. The count is incremented by one for every hold operation that is processed for the thread. It is decremented by one for every release operation. If the count is greater than 0, the thread was already held. Hold and release operations that have not completed are not reflected in the count.

Format of job or thread identification information

The format of the information needed to identify the thread for which the specified action will be taken.

JIDF0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions for JIDF0100 Format

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

*   The job in which this program is running. The job number and user name must contain blanks.

INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread identifier. A value that uniquely identifies a thread within a job. If the thread indicator is not 0, this field must contain hexadecimal zeros.

Thread indicator. A value that is used to specify the thread within the job for which the action is to be taken. The following values are supported:

0   Action should be taken for the thread specified in the thread identifier field.

1   Action should be taken for the thread that this program is running in currently. The combination of the internal job identifier, job name, job number, and user name fields also must identify the job containing the current thread.

2   Action should be taken for the initial thread of the identified job.

User name. A specific user profile name, or blanks when the job name specified is a special value.

JIDF0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	UNSIGNED BINARY(4)	Thread handle
48	30	CHAR(8)	Thread identifier

Field Descriptions for JIDF0200 Format

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

* The job in which this program is running. The job number and user name must contain blanks.

INT The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread handle. A value that is used to address a particular thread within a job. While the thread identifier uniquely identifies the thread within the job, the thread handle can imporve performance when referencing the thread. A valid thread handle must be specified. The thread handle is returned on several other interfaces.

Thread identifier. A value that uniquely identifies a thread within a job. A valid thread identifier must be specified.

User name. A specific user profile name, or blanks when the job name specified is a special value.

Error Messages

Message ID	Error Message Text
CPF1071 E	No authority to job &3/&2/&1.
CPF136A E	Job &3/&2/&1 not active.

CPF18BF E	Thread &1 not found.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
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.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C3C E	Value for parameter &1 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.
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.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFB431 E	Ending the initial thread is not allowed.
CPFB438 E	Holding the initial thread of a system job is not allowed.
CPFB439 E	Releasing the initial thread of a system job is not allowed.

《

API introduced: V5R2

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.

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

Create Job Structures (QWTCTJBS) API

Required Parameter Group:

1	Current number of temporary job structures available	Output	Binary(4)
2	Current number of permanent job structures available	Output	Binary(4)
3	Number of temporary job structures to create	Input	Binary(4)
4	Error Code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Create Job Structures (QWTCTJBS) API creates the number of temporary job structures that are passed on the call. The current number of temporary and permanent job structures available are returned.

Authorities and Locks

Job Authority

> *JOBCTL

Required Parameter Group

Current number of temporary job structures available

> OUTPUT; BINARY(4)

> The number of temporary job structures that currently exist on the system that are not in use. This number also includes any temporary job structures that are to be created on this call to the API.

Current number of permanent job structures available

> OUTPUT; BINARY(4)

> The number of permanent job structures that currently exist on the system that are not in use.

Number of temporary job structures to create

> INPUT; BINARY(4)

> The number of additional temporary job structures that the user would like to have created. The valid range is 0-32000. If a number outside that range is passed, an error will be signaled. The following special value can be passed:

> *0* No additional temporary job structures are created. The current number of temporary and permanent job structures are returned.

Error code

> I/O; CHAR(*)

> The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Error Messages

Message ID	Error Message Text
CPE3002 E	A range error occurred.
CPF222E E	&1 special authority is required.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C20 E	Error found by program &1.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R1

Delete Job Structures (QWTDTJBS) API

```
Required Parameter Group:

  1    Number of temporary job structures      Input      Binary(4)
       to delete
  2    Error code                               I/O        Char(*)

Default Public Authority: *EXCLUDE

Threadsafe: No
```

The Delete Job Structures (QWTDTJBS) API deletes the number of temporary job structures that are passed on the call. The Create Job Structures (QWTCTJBS) API can be used to create temporary job structures and retrieve the current number available.

Authorities and Locks

Job Authority
> *JOBCTL

Required Parameter Group

Number of temporary job structures to delete
> INPUT; BINARY(4)

> The number of temporary job structures that the user would like to have deleted. The valid range is 1 through 32000. If a number outside that range is passed, an error will be signaled. If there are fewer temporary job structures available than the number requested to be deleted, all available job structures will be deleted and no error is signaled.

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
CPE3002 E	A range error occurred.
CPF222E E	&1 special authority is required.

CPF24B4 E   Severe error while addressing parameter list.

CPF3CF1 E   Error code parameter not valid.

CPF3CF2 E   Error(s) occurred during running of &1 API.

CPF3C20 E   Error found by program &1.

CPF3C36 E   Number of parameters, &1, entered for this API was not valid.

CPF3C90 E   Literal value cannot be changed.

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

API introduced: V4R4

Dump Flight Recorder (QWTDMPFR) API

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 following types of jobs have flight recorders:

- Subsystem monitors
- System jobs

The QWTDMPFR API has no parameters.

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

API introduced: V2R2

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: *EXCLUDE

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.

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.
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

List Active Subsystems (QWCLASBS) API

```
Required Parameter Group:

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


Default Public Authority: *USE

Threadsafe: No
```

The List Active Subsystems (QWCLASBS) API retrieves a list of active subsystems. QWCLASBS replaces any existing data. It does not add the new list to an existing one. To retrieve more information about active subsystems, see Retrieve Subsystem Information (QWDRSBSD) API.

Authorities and Locks

User Space Authority
> *CHANGE

Library Authority
> *EXECUTE

User Space Lock
> *EXCLRD

Required Parameter Group

Qualified user space name
> INPUT; CHAR(20)
>
> The user space that receives the list, and the library in which it is located. The first 10 characters contain the user space name. The second 10 characters contain the library name. You can use these special values for the library name:
>
> *CURLIB*   The job's current library
>
> *LIBL*   The library list

Format name
> INPUT; CHAR(8)
>
> The format to use for the list of active subsystems. You can use this format name:

SBSL0100   Basic subsystem list. See [Format of the Generated List](#).

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Format of the Generated List

The list of active subsystems that the QWCLASBS API returns into the user space consists of:

- A user area
- A generic header
- An input parameter section
- A list data section

The user area and generic header are described in [User Space Format for List APIs](#). The remaining items are described in the following sections. For detailed descriptions of the fields in the tables, see [Field Descriptions](#).

Input Parameter Section

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library specified
20	14	CHAR(8)	Format name specified

SBSL0100 Format

This section is repeated for each active subsystem.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Subsystem description name
10	A	CHAR(10)	Subsystem description library name

Field Descriptions

Format name specified. The format name as specified in the call to the API.

Subsystem description library name. The name of the library in which the active subsystem description resides.

Subsystem description name. The name of the active subsystem about which information is being returned.

User space library specified. The library name or special value specified in the call to this API.

User space name. The name of the user space that receives the list.

Error Messages

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
≫ CPF3CF2 E	Error(s) occurred during running of &1 API.≪
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

List Job (QUSLJOB) API

The List Job (QUSLJOB) API generates a list of all or some jobs on the system. The generated list replaces any existing list in the user space.

The QUSLJOB API produces a list similar to the list produced by the Work with User Job (WRKUSRJOB) command.

Authorities and Locks

User Space Authority

   *CHANGE

Library Authority

   *EXECUTE

User Space Lock

*EXCLRD

Job Authority

If format JOBL0200 is specified, then for each job for which information is retrieved, the caller of the API must be running under a user profile that is the same as the job user identity of the job for which the information is being retrieved. Otherwise, the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the [Work Management](#) book on the V5R1 Supplemental Manuals Web site.

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The user space that is to receive the generated list, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name. You can use these special values for the library name:

CURLIB	The job's current library
LIBL	The library list

Format name

INPUT; CHAR(8)

The format of the job list to be returned. If format JOBL0200 is specified, the fields that were selected by the caller will be returned for each job in the list. This format is slower than the JOBL0100 format. The performance will vary depending on the number of fields selected.

You must use one of the following format names:

JOBL0100	Basic job list.
JOBL0200	Basic job list with keyed return fields.

For more information, see [Format of the Generated List](#).

Qualified job name

INPUT; CHAR(26)

The name of the job to be included in the list. The qualified job name has three parts:

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

	*	Only the job that this program is running in. The rest of the qualified job name parameter must be blank.
	CURRENT	All jobs with the current job's name.
	ALL	All jobs. The rest of the job name parameter must be specified.

User name	CHAR(10). A specific user profile name, a generic name, or one of the following special values:

	CURRENT	Jobs with the current job's user profile.
	ALL	Jobs with the specified job name, regardless of the user name. The rest of the job name parameter must be specified.

Job number	CHAR(6). A specific job number or the following special value:

	ALL	Jobs with the specified job name and user name, regardless of the job number. The rest of the job name parameter must be specified.

Status

INPUT; CHAR(10)

The status of the jobs to be included in the list. The special values supported are:

ACTIVE	Active jobs. This includes group jobs, system request jobs, and disconnected jobs.
JOBQ	Jobs currently on job queues.
OUTQ	Jobs that have completed running but still have output on an output queue.
ALL	All jobs, regardless of status.

Optional Parameter Group 1

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.

Optional Parameter Group 2

Job type

INPUT; CHAR(1)

The type of job to be listed. Refer to [Comparing Job Type and Subtype with the Work with Active Job Command](#) in the Retrieve Job Information (QUSRJOBI) API for information about how the job type field and the job subtype field equate to the type field in the Work with Active Job (WRKACTJOB) command.

The possible values for this parameter are:

*	This value lists all job types.
A	The job is an autostart job.
B	The job is a batch job.
I	The job is an interactive job.
M	The job is a subsystem monitor job.
R	The job is a spooled reader job.
S	The job is a system job.
W	The job is a spooled writer job.
X	The job is the start-control-program-function (SCPF) system job.

Number of fields to return

INPUT; BINARY(4)

The number of fields to return in the JOBL0200 format. This parameter is only used for the JOBL0200 format. If JOBL0100 is specified for the format name, the value must be zero.

Key of fields to be returned

INPUT; ARRAY(*) of BINARY(4)

The list of the fields to be returned in the JOBL0200 format. For a list of the valid fields, see [Valid Keys](#). This parameter is used for the JOBL0200 format only. If JOBL0100 is specified for the format name, the value must be zero.

Optional Parameter Group 3

Continuation handle

INPUT; CHAR(48)

The value returned to the user in the header section when a partial list is returned. The possible values are:

blank	This starts at the beginning of the list. This value is used if this parameter is omitted.

value The entries after this value matching the job name specified are returned in the list. For more information on using this value to process a partial list, see Partial List Considerations.

Format of the Generated List

The job list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see Field Descriptions.

When you retrieve list entry information from a user space, you should use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name specified
10	A	CHAR(10)	User name specified
20	14	CHAR(6)	Job number specified
26	1A	CHAR(10)	Status
36	24	CHAR(10)	User space specified
46	2E	CHAR(10)	User space library specified
56	38	CHAR(8)	Format name specified
64	40	CHAR(1)	Job type specified
65	41	CHAR(3)	Reserved
68	44	BINARY(4)	Number of fields to return specified
72	48	ARRAY(*) of BINARY(4)	Key of fields to return specified
*	*	CHAR(48)	Continuation handle

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name used
10	A	CHAR(10)	User name used
20	14	CHAR(6)	Job number used
26	1A	CHAR(48)	Continuation handle

JOBL0100 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name used
10	A	CHAR(10)	User name used
20	14	CHAR(6)	Job number used
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(10)	Status
52	34	CHAR(1)	Job type
53	35	CHAR(1)	Job subtype
54	36	CHAR(2)	Reserved

JOBL0200 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(56)	Everything in JOBL0100 format
56	38	CHAR(1)	Job information status
57	39	CHAR(3)	Reserved
60	3C	BINARY(4)	Number of fields returned
These fields repeat, in the order listed, for each key selected.		BINARY(4)	Length of field information returned
		BINARY(4)	Key field
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data
		CHAR(*)	Data
		CHAR(*)	Reserved

Field Descriptions

Data. The data returned for the key field.

Format name specified. The format name as specified in the call to the API.

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

Job information status. Whether the information was available for the job. The possible values are:

blank   The information was available.

A   The user was not authorized to the job.

L   The information was not available because the job was not accessible.

Job name specified. The name of the job as specified in the call to the API.

Job name used. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the work station where the job started; for a batch job, you specify the name in the command when you submit the job.

Job number specified. The job number as specified in the call to the API.

Job number used. The system-assigned job number.

Job subtype. Additional information about the job type (if any exists). Refer to Comparing Job Type and Subtype with the Work with Active Job Command in the Retrieve Job Information (QUSRJOBI) API for information about how the job type field and the job subtype field equate to the type field in the Work with Active Job (WRKACTJOB) command. The possible values are:

blank   The job has no special subtype.

D   The job is a batch immediate job.

E   The job started with a procedure start request.

F   The job is an AS/400 Advanced 36 machine server job.

J   The job is a prestart job.

P   The job is a print driver job.

T   The job is a System/36 multiple requester terminal (MRT) job.

U   The job is an alternate spool user.

Job type. The type of job. Refer to Comparing Job Type and Subtype with the Work with Active Job Command in the Retrieve Job Information (QUSRJOBI) API for information about how the job type field and the job subtype field equate to the type field in the Work with Active Job (WRKACTJOB) command. The possible values for this field are:

A   The job is an autostart job.

B   The job is a batch job.

I   The job is an interactive job.

M   The job is a subsystem monitor job.

R   The job is a spooled reader job.

S   The job is a system job.

W   The job is a spooled writer job.

X   The job is the SCPF system job.

Job type specified. The job type as specified in the call to the API.

Key field. The field returned. See Valid Keys for the list of valid keys.

Key of fields to return specified. The key of fields to return as specified in the call to the API.

Length of data. The length of the data returned for the field.

Length of field information returned. The total length of information returned for this field. This value is used to increment to the next field in the list.

Number of fields returned. The number of fields returned to the application.

Number of fields to return specified. The number of fields to return as specified in the call to the API.

Reserved. An ignored field.

Status. The status of the job. The valid values are:

*ACTIVE   The job has started, and it can use system resources (processing unit, main storage, and so on). This does not guarantee that the job is currently running, however. For example, an active job may be in one of the following states where it is not in a position to use system resources:

- The Hold Job (HLDJOB) command holds the job; the Release the (RLSJOB) command allows the job to run again.
- The Transfer Group Job (TFRGRPJOB) or Transfer Secondary Job (TFRSECJOB) command suspends the job. When control returns to the job, the job can run again.
- The job is disconnected using the Disconnect Job (DSCJOB) command. When the interactive user signs back on, thereby connecting back into the job, the job can run again.
- The job is waiting for any reason. For example, when the job receives the reply for an inquiry message, the job can start running again.

*JOBQ   The job is currently on a job queue. The job possibly was previously active and was placed back on the job queue because of the Transfer Job (TFRJOB) or Transfer Batch Job (TFRBCHJOB) command, or the job was never active because it was just submitted.

*OUTQ   The job has completed running and has spooled output that has not yet printed.

Type of data. The type of data returned.

C   The data is returned in character format.

B   The data is returned in binary format.

User name specified. The user name as specified in the call to the API.

User name used. The user profile under which the job is run. The user name is the same as the user profile name and can come from several different sources depending on the type of job.

User space library specified. The name of the library containing the user space as specified in the call to the API.

User space specified. The name of the user space as specified in the call to the API.

Valid Keys

The following table contains a list of the valid keys. The descriptions of all the valid key attributes are described in Work Management API Attribute Descriptions (WMAttrDesc).

Key	Type	Description
0101	CHAR(4)	Active job status
0102	CHAR(1)	Allow multiple threads
0103	CHAR(4)	Active job status for jobs ending
0201	CHAR(10)	Break message handling
0301	CHAR(1)	Cancel key
0302	BINARY(4)	Coded character set ID
0303	CHAR(2)	Country or region ID
0304	BINARY(4)	Processing unit time used, if less than 2,147,483,647 milliseconds
0305	CHAR(10)	Current user profile
0306	CHAR(1)	Completion status
0307	BINARY(4)	Current system pool identifier
0311	CHAR(10)	Character identifier control
0312	BINARY(8), UNSIGNED	Processing unit time used - total for the job
0313	BINARY(8), UNSIGNED	Processing unit time used for database - total for the job
0401	CHAR(13)	Date and time job became active
0402	CHAR(13)	Date and time job entered system
0403	CHAR(8)	Date and time job is scheduled to run
0404	CHAR(8)	Date and time job was put on this job queue
0405	CHAR(4)	Date format
0406	CHAR(1)	Date separator
0407	CHAR(1)	DBCS-capable
0408	CHAR(10)	DDM conversation handling
0409	BINARY(4)	Default wait
0410	CHAR(13)	Device recovery action
0411	CHAR(10)	Device name

0412	BINARY(4)	Default coded character set identifier
0413	CHAR(1)	Decimal format
0418	CHAR(13)	Date and time job ended
0501	BINARY(4)	End severity
0502	CHAR(1)	End status
0503	CHAR(1)	Exit key
0601	CHAR(10)	Function name
0602	CHAR(1)	Function type
0701	CHAR(1)	Signed-on job
0702	CHAR(10)	Group profile name
0703	CHAR(150)	Group profile name - supplemental
0901	CHAR(10)	Inquiry message reply
1001	CHAR(15)	Job accounting code
1002	CHAR(7)	Job date
1003	CHAR(20)	Job description name - qualified
1004	CHAR(20)	Job queue name - qualified
1005	CHAR(2)	Job queue priority
1006	CHAR(8)	Job switches
1007	CHAR(10)	Job message queue full action
1008	BINARY(4)	Job message queue maximum size
1012	CHAR(10)	Job user identity
1013	CHAR(1)	Job user identity setting
1014	BINARY(4)	Job end reason
1015	CHAR(1)	Job log pending
1016	BINARY(4)	Job type - enhanced
1201	CHAR(3)	Language ID
1202	CHAR(1)	Logging level
1203	CHAR(10)	Logging of CL programs
1204	BINARY(4)	Logging severity
1205	CHAR(10)	Logging text
1301	CHAR(8)	Mode name
1302	BINARY(4)	Maximum processing unit time
1303	BINARY(4)	Maximum temporary storage in kilobytes
1304	BINARY(4)	Maximum threads
1305	BINARY(4)	Maximum temporary storage in megabytes
1306	CHAR(10)	Memory pool name
1307	CHAR(1)	Message reply
1401	BINARY(4)	Number of auxiliary I/O requests, if less than 2,147,483,647
1402	BINARY(4)	Number of interactive transactions
1403	BINARY(4)	Number of database lock waits
1404	BINARY(4)	Number of internal machine lock waits
1405	BINARY(4)	Number of nondatabase lock waits

1406	BINARY(8), UNSIGNED	Number of auxiliary I/O requests
1501	CHAR(20)	Output queue name - qualified
1502	CHAR(2)	Output queue priority
1601	CHAR(10)	Print key format
1602	CHAR(30)	Print text
1603	CHAR(10)	Printer device name
1604	CHAR(10)	Purge
1605	BINARY(4)	Product return code
1606	BINARY(4)	Program return code
1607	CHAR(8)	Pending signal set
1608	BINARY(4)	Process ID number
1801	BINARY(4)	Response time total
1802	BINARY(4)	Run priority (job)
1803	CHAR(80)	Routing data
1901	CHAR(20)	Sort sequence table - qualified
1902	CHAR(10)	Status message handling
1903	CHAR(10)	Status of job on the job queue
1904	CHAR(26)	Submitter's job name - qualified
1905	CHAR(20)	Submitter's message queue name - qualified
1906	CHAR(20)	Subsystem description name - qualified
1907	BINARY(4)	System pool identifier
1908	CHAR(10)	Special environment
1909	CHAR(8)	Signal blocking mask
1910	BINARY(4)	Signal status
1911	CHAR(30)	Server type
» 1982	CHAR(10)	Spooled file action«
2001	CHAR(1)	Time separator
2002	BINARY(4)	Time slice
2003	CHAR(10)	Time-slice end pool
2004	BINARY(4)	Temporary storage used in kilobytes
2005	CHAR(3)	Time spent on database lock waits
2006	CHAR(3)	Time spent on internal machine lock waits
2007	CHAR(3)	Time spent on nondatabase lock waits
2008	BINARY(4)	Thread count
2009	BINARY(4)	Temporary storage used in megabytes
2101	CHAR(24)	Unit of work ID
2102	BINARY(4)	User return code

Usage Notes

The conditions under which this API is threadsafe are the same as those described in the Usage Notes for the Retrieve Job Information (QUSRJOBI) API.

Error Messages

Message ID	Error Message Text
CPF1865 E	Value &1 for job type not valid.
CPF1866 E	Value &1 for number of fields to return not valid.
CPF1867 E	Value &1 in list not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CB1 E	Value &1 for job status is not valid.
CPF3CB2 E	Value specified for job parameter is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
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.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V1R3

List Job Schedule Entries (QWCLSCDE) API

```
Required Parameter Group:

    1    Qualified user space name      Input      Char(20)
    2    Format name                    Input      Char(8)
    3    Job schedule entry name        Input      Char(10)
    4    Continuation handle            Input      Char(16)
    5    Error code                     I/O        Char(*)

Default Public Authority: *USE

Threadsafe: No
```

The List Job Schedule Entries (QWCLSCDE) API lists the entries in the job schedule, QDFTJOBSCD. A subset of the list can be created by using the job schedule entry name parameter. The generated list replaces any existing list in the user space.

The QWCLSCDE API produces a list similar to the list produced by the Work with Job Schedule Entries (WRKJOBSCDE) command.

Authorities and Locks

User Space Authority
>   *CHANGE
User Space Library Authority
>   *EXECUTE
User Space Lock
>   *EXCLRD
Job Schedule Entry Authority
>   *USE if using format SCDL0100; *JOBCTL or the adder of the entry if using format SCDL0200
Job Schedule Authority
>   *USE
Job Schedule Library Authority
>   ≫*EXECUTE≪
Job Schedule Lock
>   *SHRRD

Required Parameter Group

Qualified user space name

    INPUT; CHAR(20)

    The user space that is to receive the created list. The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

    CURLIB   The job's current library

    LIBL   The library list

Format name

    INPUT; CHAR(8)

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

    SCDL0100   Basic job schedule entries list.

    SCDL0200   Detailed job schedule entries list. This format requires more processing than the SCDL0100 format.

    For more information, see [SCDL0100 Format](#) or [SCDL0200 Format](#).

Job schedule entry name

    INPUT; CHAR(10)

    The job schedule entry about which to retrieve information. This can be used to create a subset of job schedule entries by using the following values:

    ALL   All of the job schedule entries are returned in the list.

    generic*   All of the job schedule entries beginning with the generic value are returned in the list.

    name   The job schedule entries with the given name are returned in the list.

Continuation handle

    INPUT; CHAR(16)

    The value returned to the user in the header section when a partial list is returned. The possible values are:

    blank   This will start at the beginning of the list.

    value   The entries after this value matching the job schedule entry name specified will be returned in the list. For more information on using this value to process a partial list, see [Partial list considerations](#).

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error code parameter](#).

Format of the Generated Lists

The file member list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - SCDL0100 format
  - SCDL0200 format

For details about the user area and generic header, see [User space format for list APIs](#). For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see [Field Descriptions](#).

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API [Examples](#).

Input Parameter Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library name
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	Job schedule entry name
44	2C	CHAR(16)	Continuation handle

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job schedule entry name used
10	A	CHAR(16)	Continuation handle

SCDL0100 Format

Offset Dec	Offset Hex	Type	Field
0	0	CHAR(1)	Information status
1	1	CHAR(10)	Job name
11	B	CHAR(10)	Entry number
21	15	CHAR(10)	Scheduled date
31	1F	CHAR(70)	Scheduled days
101	65	CHAR(6)	Scheduled time
107	6B	CHAR(10)	Frequency
117	75	ARRAY(5) of CHAR(10)	Relative day of the month
167	A7	CHAR(10)	Recovery action
177	B1	CHAR(10)	Next submission date
187	BB	CHAR(10)	Status
197	C5	CHAR(10)	Job queue name
207	CF	CHAR(10)	Job queue library name
217	D9	CHAR(10)	User profile of entry adder
227	E3	CHAR(10)	Last submission date
237	ED	CHAR(6)	Last submission time
243	F3	CHAR(50)	Text
293	125	CHAR(23)	Reserved

SCDL0200 Format

Offset Dec	Offset Hex	Type	Field
0	0		Returns everything from format SCDL0100
316	13C	CHAR(10)	Job queue status
326	146	ARRAY(20) of CHAR(10)	Dates omitted
526	20E	CHAR(10)	Job description name
536	218	CHAR(10)	Job description library name
546	222	CHAR(10)	User profile for submitted job
556	22C	CHAR(10)	Message queue name
566	236	CHAR(10)	Message queue library name
576	240	CHAR(10)	Save entry
586	24A	CHAR(10)	Last submission job name

596	254	CHAR(10)	Last submission user name
606	25E	CHAR(6)	Last submission job number
612	26E	CHAR(10)	Last attempted submission date
622	26E	CHAR(6)	Last attempted submission time
628	274	CHAR(10)	Status of last attempted submission
638	27E	CHAR(2)	Reserved
640	280	BINARY(4)	Length of command string
644	284	CHAR(512)	Command

Field Descriptions

Command. A command that runs in the submitted batch job if the routing program used when this batch job is started is the IBM-supplied default routing program QCMD. Because this command is used for the request data, this parameter takes the place of any value specified for the RQSDTA parameter in the job description.

Continuation handle. The value used to process a partial list. This value is put in the header section when a partial list is returned. For more information on processing a partial list, see Partial list considerations.

Dates omitted. Specifies up to 20 dates when the job should not be submitted. The dates will be in the format CYYMMDD, where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*. If no omit dates are specified, this field contains hexadecimal zeros.

Entry number. The entry number assigned to the job schedule entry. The entry number can range from 000001 through 999999.

Format name. The content and format of the information returned for each job schedule entry.

Frequency. How often the job schedule entry is to be submitted to run. Valid values are:

ONCE   The job schedule entry does not repeat.

WEEKLY   The job schedule entry is submitted on the same day of each week at the scheduled time.

MONTHLY   The job schedule entry is submitted on the same day of each month at the scheduled time.

Information status. Whether or not the entry information could be successfully retrieved.

blank   No errors occurred. All information is returned for this entry.

A   Insufficient authority to entry. Only the SCDL0100 information is returned for this entry. The rest of the entry is blank.

L   The entry is locked. Only the SCDL0100 information is returned for this entry. The rest of the entry is blank.

Job description name. The name of the job description used for the job schedule entry.

USRPRF   The job description in the user profile under which the job runs is used as the job description of the job schedule entry.

job description name   The name of the job description used for the job schedule entry.

Job description library name. The name of the library in which the job description is located.

Job name. The job name associated with the job schedule entry.

Job queue library name. The name of the library in which the specified job queue resides.

Job queue name. The name of the job queue where the job should be placed when it is submitted. Valid values are:

*JOBD   The job is placed in the job queue named in the specified job description.

job queue name   The name of the job queue where the job is placed when it is submitted.

Job queue status. The current status of the job queue. Valid values are:

blank   The job queue name specified *JOBD, the job queue could not be found, or the job queue is damaged.

HLD   The job queue is held. The job queue is not allocated to a subsystem.

RLS   The job queue is released. The job queue is not allocated to a subsystem.

HLD/SBS   The job queue is held. The job queue is allocated to a subsystem.

RLS/SBS   The job queue is released. The job queue is allocated to a subsystem.

LOCKED   The job queue is locked, and the status could not be obtained.

Job schedule entry name used. The job name used to identify the job schedule entry.

Last attempted submission date. The date on which a job could have been last submitted. The value is returned in the format CYYMMDD, where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*. If no submission has been attempted, this field contains hexadecimal zeros.

Last attempted submission time. The time at which a job could have been last submitted from this entry. The value is returned in the format HHMMSS, where HH is hours, MM is minutes, and SS is seconds. If no submission has been attempted, this field contains hexadecimal zeros.

Last submission date. The date on which a job was last submitted from this entry. The value is returned in the format CYYMMDD, where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*. If there has been no previous submission, this field contains hexadecimal zeros.

Last submission job name. The job name of the last job that was submitted from this entry. If there has been no previous submission, this field contains blanks.

Last submission job number. The job number of the last job that was submitted from this entry. If there has been no previous submission, this field contains blanks.

Last submission time. The time at which a job was last submitted from this entry. The value is returned in the format HHMMSS, where HH is hours, MM is minutes, and SS is seconds. If there has been no previous submission, this field contains hexadecimal zeros.

Last submission user name. The user name of the last job that was submitted from this entry. If there has

been no previous submission, this field contains blanks.

Length of command string. The length of the command string specified in the command field.

Message queue name. The name of the message queue, if any, to which a completion message is sent when the job is submitted. A completion message is sent when the submitted job has completed running, and error messages are sent if the Submit Job (SBMJOB) command fails for some reason. Valid values are:

USRPRF	A completion message is sent to the message queue specified in the user profile associated with the job schedule entry.
NONE	Messages are not sent to a user-specified message queue. In this case, completion messages are not sent, but error messages are sent to the QSYSOPR message queue.
message queue name	The name of the message queue where the messages are sent.

Message queue library name. The library in which the message queue is located.

Next submission date. The next date that a job from this entry is scheduled to be submitted. The next submission date is returned in the format CYYMMDD, where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1 indicates years 20*xx*. If this entry has a status of SAV, this field contains hexadecimal zeros.

Recovery action. The action that will happen if the job cannot be submitted at the designated time because the system is powered down or the system is in the restricted state. The action specified by this parameter then occurs when the system is IPLed or when the system comes out of the restricted state. This parameter does not pertain to the situation where a job was held (by the user) when the designated time elapsed and then released (by the user) at a later time. Also, the recovery action does not pertain to timer events that elapse as a result of changes to the QTIME system value. Valid values are:

SBMRLS	Submit the job in the released state (RLS).
SBMHLD	Submit the job in the held state (HLD).
NOSBM	No job is submitted.

Relative day of the month. The relative day of the month the job should be submitted to run. A total of five values can be returned. If no relative day of the month was specified, this area contains blanks. Valid values are:

1-5	The job should be submitted on the specified day of the week every first, second, third, fourth, or fifth week of the month.
LAST	The job should be submitted on the last specified day of the week each month.

Reserved. An ignored field.

Save entry. Whether or not an entry that has FRQ(*ONCE) specified should be kept in the job schedule after the job has been submitted.

NO	This entry will not be kept after the job is submitted.
YES	This entry will be kept after the job is submitted.

Scheduled date. The date that the job will be submitted. Valid values are:

CURRENT   The current date will be used.

MONTHSTR   The first day of the month will be used.

MONTHEND   The last day of the month will be used.

NONE   A scheduled date was not specified.

date   An actual date in the format CYYMMDD, where C is the century, YY is the year, MM
               is the month, and DD is the day. A 0 for the century flag indicates years 19*xx* and a 1
               indicates years 20*xx*.

Scheduled days. The day of the week that the job will be submitted. A total of seven values can be
returned. Valid values are:

ALL   The job will be submitted every day. This cannot be specified with any other values.

NONE   A scheduled day is not specified. This cannot be specified with any other values.

MON   The job will be submitted on Monday.

TUE   The job will be submitted on Tuesday.

WED   The job will be submitted on Wednesday.

THU   The job will be submitted on Thursday.

FRI   The job will be submitted on Friday.

SAT   The job will be submitted on Saturday.

SUN   The job will be submitted on Sunday.

Scheduled time. The time (on the scheduled date) when the job will be submitted to run. This value is
returned in the format HHMMSS, where HH is the hours, MM is the minutes, and SS is the seconds.

Status. The status of the job schedule entry. Valid values are:

SCD   The entry is scheduled.

HLD   The entry is held.

SAV   The entry is saved.

Status of last attempted submission. The action that occurred the last time the system could have
submitted a job from this entry. Valid values are:

0   Job not previously submitted.

1   Job successfully submitted.

2   Last job submission failed. Check the message queue for details.

3   Job not submitted due to held status.

4   Job submitted after scheduled time as specified by recovery action.

5   Job not submitted as specified by recovery action.

Text. Text that briefly describes the job schedule entry. If no text was specified, this field contains blanks.

User profile for submitted job. The user profile under which the job will be submitted. Valid values are:

*JOBD   The user profile named in the specified job description is used for the job.

user name   The name of the user profile that is used for the job.

User profile of entry adder. The user profile that created this entry.

User space library name. The library name or special value specified in the call to this API.

User space name. The name of the user space that receives the list.

Error Messages

Message ID	Error Message Text
CPF1629 E	Not authorized to job schedule &1.
CPF1632 E	Job schedule entry &3 number &4 damaged.
CPF1637 E	Job schedule &1 in library &2 in use.
CPF1640 E	Job schedule &1 in library &2 does not exist.
CPF1641 E	Job schedule &1 in library &2 damaged.
CPF1643 E	Job schedule entry name not valid.
CPF3CF1 E	Error code parameter not valid.
» CPF3CF2 E	Error(s) occurred during running of &1 API. «
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF811A E	User space &4 in &9 damaged.
CPF812C E	Job schedule &4 in &9 damaged.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.

CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

List Object Locks (QWCLOBJL) API

Required Parameter Group:

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

Optional Parameter Group 1:

7	Path name	Input	Char(*)
8	Path name length	Input	Binary(4)

≫Optional Parameter Group 2:

9	Qualified object ASP name	Input	Char(10)≪

Default Public Authority: *USE

Threadsafe: No

The List Object Locks (QWCLOBJL) API generates a list of lock information about a specific object or database file member and places the list into the specified user space. An object level or member level lock may be specified. If it is a database file, you will get a lock on the member (if requested); otherwise, the lock is on the object. This API provides information similar to that provided by the Work with Object Lock (WRKOBJLCK) command.

Authorities and Locks

User Space Authority
  *CHANGE
User Space Library Authority
  *EXECUTE
User Space Lock
  *EXCLRD
≫*ASP device*
  *EXECUTE
Object library
  *EXECUTE

A user with *JOBCTL special authority is not required to have *EXECUTE authority to either the auxiliary storage pool (ASP) device or the library containing the object.

If a path name is specified, *X authority is required for directories in the path reqardless of any special authorities the user may have.≪

Required Parameter Group

Qualified user space name

      INPUT; CHAR(20)

      The name of the existing user space that is to receive the created list. The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

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

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

Format name

      INPUT; CHAR(8)

      The name of the format used to list object locks. You can specify this format:

      OBJL0100   The content and format of the lock information being returned. For more information, see [OBJL0100 Format](#).

Qualified object name

      INPUT; CHAR(20)

      The name of the object whose locks are to be placed in the list. The first 10 characters contain the object name, and the second 10 characters contain the name of the library where the object is located.

      If you want to use a path name instead of a qualified object name, use this special value for the object name:

      *OBJPATH   Use the optional parameters, path name and path name length, to specify the object name. When this special value is specified, the member name field must be the special value *NONE and the Object type field must be blanks.

      You can use these special values for the library name:

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

      *LIBL   The library list is used to locate the object.

Object type

INPUT; CHAR(10)

The object type of operating system object for which the list of locks is returned. Specify the predefined value that identifies the object type. See the CL Programming book for more information on allowed object types. If a path name has been specified, then this field must contain blanks.

Member name

INPUT; CHAR(10)

This parameter is valid only when a database file has been specified in the qualified object name parameter. For other than database files, use *NONE. Possible values are a specific name or a special value:

NONE   No member locks are retrieved, but file level locks are retrieved. If the qualified object name is not a database file, use this value. If a path name is being specified for the object name, use this value.

FIRST   The member locks for the first member in the named file are retrieved.

ALL   Member locks for all the members in the file are retrieved.

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

Path name

INPUT; CHAR(*)

The path name of the object whose locks are to be placed in the list. Both absolute and relative path names are allowed. The patterns ? and * are not allowed. The home directory of the user is not resolved, thus a tilde (~) in the first character position is not treated as the home directory. This parameter is assumed to be represented in the coded character set identifier (CCSID) currently in effect for the job. If the CCSID of the job is 65535, this parameter is assumed to be represented in the default CCSID of the job. The path name delimiter must be a slash (/). If a symbolic link is specified, the link is not followed.

Path name length

INPUT; BINARY(4)

The length of the path name, in bytes.

≫Optional Parameter Group 2

Qualified object ASP name

      INPUT; CHAR(10)

      The name of the ASP device where the object's library is located. This parameter must be * if the library portion of the qualified object name is *CURLIB or *LIBL. It also must be * if the qualified object name is *OBJPATH. If the object is a library and either an ASP device name or *SYSBAS is specified, the library portion of the qualified object name must be QSYS. The following special values may be specified:

      *   The ASPs that are currently part of the thread's library name space will be searched to locate the object.

      SYSBAS   The system ASP and all basic ASPs will be searched to locate the object.≪

Format of the Generated List

The file member list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section

For details about the user area and generic header, see [User Space Format for List APIs](). For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see [Field Descriptions]().

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API [Examples]().

Input Parameter Section

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	CHAR(10)	Object name specified
38	26	CHAR(10)	Object library name specified
48	30	CHAR(10)	Object type specified
58	3A	CHAR(10)	Member name specified
68	44	BINARY(4)	Offset to path name specified

Dec	Hex	Type	Field
72	48	BINARY(4)	Length of path name specified
» 76	4C	CHAR(10)	Object library ASP name specified«
		CHAR(*)	Path name specified

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used
20	14	CHAR(10)	Object name used
30	1E	CHAR(10)	Object library name used
40	28	CHAR(10)	Object type returned
50	32	CHAR(10)	Extended object attribute returned
60	3C	CHAR(10)	Shared file name
70	46	CHAR(10)	Shared file library name
80	50	BINARY(4)	Offset to path name used
84	54	BINARY(4)	Length of path name used
» 88	58	CHAR(10)	Object ASP name used
98	62	CHAR(10)	Object library ASP name used«
		CHAR(*)	Path name used

OBJL0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	Job user name
20	14	CHAR(6)	Job number
26	1A	CHAR(10)	Lock state
36	24	BINARY(4)	Lock status
40	28	BINARY(4)	Lock type
44	2C	CHAR(10)	Member name
54	36	CHAR(1)	Share
55	37	CHAR(1)	Lock scope
56	38	CHAR(8)	Thread identifier

Field Descriptions

Extended object attribute returned. The extended attribute of the object for which the list of locks is returned, such as a program or a file type. Extended attributes further describe the object. For example, an object type of *FILE may have an extended object attribute of PHY (physical file), LGL (logical file), DSP (display file), SAV (save file), and so forth.

Format name specified. The name of the format used to list object locks.

Job name. The simple job name of the job that issued the lock request. »The following special values also may be returned:

MACHINE   The lock is held by an internal machine process. If this value is returned, the job number and job user name will be blank.

*LCKSPC   The lock is attached to a lock space. If this value is returned, the job number and job user name will be blank.

*N   The job name cannot be determined.«

Job number. The system-assigned job number of the job that issued the lock request. »The following special value also may be returned:

*N   The job number cannot be determined.«

Job user name. The user name under which the job that issued the lock request is run. The user name is the same as the user profile name and can come from several different sources depending on the type of job. »The following special value also may be returned:

*N   The job user name cannot be determined.«

Length of path name specified. The length, in bytes, of the path name of the object that is specified on the call to the API.

Length of path name used. The length, in bytes, of the path name of the object for which the locks are placed in the list.

Lock scope. The scope of the lock. »The possible values are:

0   Job scope

1   Thread scope

2   Lock space scope«

Lock state. The lock condition for the lock request. The possible values are:

*NONE   No lock exists.

*SHRRD   Lock shared for read.

*SHRUPD   Lock shared for update.

*SHRNUP   Lock shared no update.

*EXCLRD   Lock exclusive allow read.

EXCL   Lock exclusive no read.

Lock status. The status of the lock. The lock may be a single request or part of a multiple lock request for which some other object specified in the request has been identified as unavailable. The possible values are:

1   The lock is currently held by the job or thread.

2   The job or thread is waiting for the lock (synchronous).

3   The job or thread has a lock request outstanding for the object (asynchronous).

Lock type. The lock type to be processed. The possible values are:

1   Lock on the object

2   Lock on the member control block

3   Lock on the access path used to access a member's data

4   Lock on the actual data within the member

Member name. The name of the file member for which the lock was requested. This field is blank if not applicable to object type.

Member name specified. The member name of a database file specified on the call to the API.

≫**Object ASP name used.** The name of the ASP device that contains the object for which the locks are placed in the list. The following special value also may be returned:

SYSBAS   The object is located in the system ASP or a basic user ASP.

Object library ASP name specified. The name of the ASP device that contains the object specified on the call to the API. The following special values may also be returned:

*   The ASPs that are currently part of the thread's library name space will be searched to locate the object.

SYSBAS   The object is located in the system ASP or a basic user ASP.

Object library ASP name used. The name of the ASP device that contains the library of the object for which locks are placed in the list. The following special value may also be returned:

SYSBAS   The library is located in the system ASP or a basic user ASP.≪

Object library name specified. The name of the library that contains the object specified on the call to the API. This field is blank if a path name was specified as the object name.

Object library name used. The name of the library that contains the object whose locks are placed in the list. This field is blank if a path name was specified as the object name.

Object name specified. The name of the object specified on the call to the API. This field will contain the special value *OBJPATH if a path name is specified.

Object name used. The name of the object for which the locks are placed in the list. This field will contain the special value *OBJPATH if a path name is specified.

Object type returned. The type of object for which locks are retrieved. This field will contain blanks if a path name is specified.

Object type specified. The type of object for which the list of locks are requested. This field will contain blanks if a path name is specified.

Offset to path name specified. The offset to the path name of the object that is specified on the call to the API.

Offset to path name used. The offset to the path name of the object for which the locks are placed in the list.

Path name specified. The path name of the object that is specified on the call to the API.

Path name used. The actual path name of the object for which the locks are placed in the list.

Share. Whether shared file member locks are associated with the file member.

0 The file is not shared, the file is a physical file, or the field is not applicable to object type.

1 The file is shared.

Shared file library name. The name of the library that contains the shared file. This field is blank if not applicable to object type or if there is no shared file. When this field has a value, it applies to all entries in the list.

Shared file name. The name of one shared file whose members are locked. This field is blank if not applicable to object type or if there is no shared file. When this field has a value, it applies to all entries in the list.

Thread identifier. The identifier of the thread that is holding a thread-scoped lock or waiting for a lock. ≫For locks that do not have a lock scope of thread scope, the hexadecimal value 00000000 is returned.≪

User space library name specified. The name of the library that contains the user space specified in the call to the API.

User space library name used. The name of the library that contains the user space into which the generated list is put.

User space name specified. The name of the user space specified in the call to the API.

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

Error Messages

Message ID	Error Message Text
CPFA0AB E	Object name not a directory.
CPFA0A3 E	Path name resolution causes looping.
CPFA0A7 E	Path name too long.
CPFA0A9 E	Object not found.

CPFA09C E	Not authorized to object.
CPF0935 E	Cannot use member name for object type &2.
CPF0951 E	QSYS only valid library for object type &2.
CPF18A0 D	Object type field not valid.
CPF18A1 D	Member name is not valid.
CPF18A2 D	Path name parameters not specified.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
» CPF3CF2 E	Error(s) occurred during running of &1 API.«
CPF3C1B E	Object identifier not valid for lock type &1.
CPF3C1C E	Lock type &1 not valid for file attribute &2.
CPF3C21 E	Format name &1 is not valid.
CPF3C31 E	Object type &1 is not valid.
» CPF3C3A E	Value for parameter &2 for API &1 not valid.«
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9820 E	Not authorized to use library &1.
» CPF9825 E	Not authorized to device &1.«
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

List Subsystem Entries (QWDLSBSE) API

```
Required Parameter Group:

    1    Qualified user space name        Input        Char(20)
    2    List format                      Input        Char(8)
    3    Qualified subsystem name         Input        Char(20)
    4    Error code                       I/O          Char(*)


Default Public Authority: *USE

Threadsafe: No
```

The List Subsystem Entries (QWDLSBSE) API lists some of the different entries in a subsystem description. See the format information for the types of entries available. QWDLSBSE replaces any data that already exists in the user space.

Other subsystem information is available through the following APIs:

  QWDRSBSD   Retrieve Subsystem Information

  QWDLSJBQ   List Job Queue Entries

Authorities and Locks

User Space Authority
    *CHANGE
User Space Library Authority
    *EXECUTE
User Space Lock
    *EXCLRD
Subsystem Description Authority
    *USE
Subsystem Description Library Authority
    *EXECUTE

Required Parameter Group

Qualified user space name
    INPUT; CHAR(20)

    The user space that receives the list, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name. You can use these special values for the library name:

CURLIB   The job's current library

LIBL   The job's library list

List format

INPUT; CHAR(8)

The format of subsystem entries to list. You can use one of the following format names:

SBSE0100   Routing entry list. For details, see [SBSE0100 Format](#).

SBSE0200   Communications entry list. For details, see [SBSE0200 Format](#).

SBSE0300   Remote locations entry list. For details, see [SBSE0300 Format](#).

SBSE0400   Autostart job entry list. For details, see [SBSE0400 Format](#).

SBSE0500   Prestart job entry list. For details, see [SBSE0500 Format](#).

SBSE0600   Workstation name entry list. For details, see [SBSE0600 Format](#).

SBSE0700   Workstation type entry list. For details, see [SBSE0700 Format](#).

Qualified subsystem name

INPUT; CHAR(20)

The subsystem description about which to retrieve information, and the library in which the subsystem description is located. The first 10 characters contain the subsystem description name, and the second 10 characters contain the library name. You can use these special values for the library name:

CURLIB   The job's current library

LIBL   The job's library list

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error code parameter](#).

Format of the Generated List

The list of entries that the QWDLSBSE API returns into the user space consists of:

- A user area
- A generic header
- An input parameter section

- A header section
- A list data section

The user area and generic header are described in [User Space Format for List APIs](#). The remaining items are described in the following sections. For detailed descriptions of the fields in the tables, see [Field Descriptions](#).

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API [Examples](#).

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	CHAR(10)	Subsystem name specified
38	26	CHAR(10)	Subsystem library name specified

Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Subsystem name used
10	A	CHAR(10)	Subsystem library name used

SBSE0100 Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Routing entry sequence number
4	4	CHAR(10)	Routing entry program name
14	E	CHAR(10)	Routing entry program library name
24	18	CHAR(10)	Routing entry class name
34	22	CHAR(10)	Routing entry class library name
44	2C	BINARY(4)	Maximum active routing steps
48	30	BINARY(4)	Routing entry pool identifier

52	34	BINARY(4)	Compare start position
56	38	CHAR(80)	Compare value

SBSE0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Device
10	A	CHAR(8)	Mode
18	12	CHAR(10)	Communication entry job description name
28	1C	CHAR(10)	Communication entry job description library name
38	26	CHAR(10)	Default user
48	30	BINARY(4)	Maximum active jobs

SBSE0300 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(8)	Remote location
8	8	CHAR(8)	Mode
16	10	CHAR(10)	Remote location entry job description name
26	1A	CHAR(10)	Remote location entry job description library name
36	24	CHAR(10)	Default user
38	26	CHAR(2)	Reserved
48	30	BINARY(4)	Maximum active jobs

SBSE0400 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Autostart job name
10	A	CHAR(10)	Autostart job description name
20	14	CHAR(10)	Autostart job description library name

SBSE0500 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Prestart job program name
10	A	CHAR(10)	Prestart job program library name
20	14	CHAR(10)	User profile name
30	1E	CHAR(1)	Start jobs
31	1F	CHAR(1)	Wait for job
32	20	BINARY(4)	Initial number of jobs
36	24	BINARY(4)	Threshold
40	28	BINARY(4)	Additional number of jobs
44	2C	BINARY(4)	Maximum number of jobs
48	30	BINARY(4)	Maximum number of uses
52	34	BINARY(4)	Pool identifier
56	38	CHAR(10)	Prestart job name
66	42	CHAR(10)	Prestart job description name
76	4C	CHAR(10)	Prestart job description library name
86	56	CHAR(2)	Reserved
88	58	CHAR(10)	First class name
98	62	CHAR(10)	First class library name
108	6C	BINARY(4)	Number of jobs to use first class
112	70	CHAR(10)	Second class name
122	7A	CHAR(10)	Second class library name
132	84	BINARY(4)	Number of jobs to use second class

SBSE0600 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Workstation name
10	A	CHAR(10)	Job description name for workstation
20	14	CHAR(10)	Job description library name for workstation
30	1E	CHAR(10)	Control job (allocation)
40	28	BINARY(4)	Maximum active jobs

SBSE0700 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Workstation type
10	A	CHAR(10)	Job description name for type
20	14	CHAR(10)	Job description library name for type
30	1E	CHAR(10)	Control job (allocation)
40	28	BINARY(4)	Maximum active jobs

Field Descriptions

Additional number of jobs. The additional number of prestart jobs that are started when the number of prestart jobs drops below the value specified for the threshold parameter.

Autostart job description library name. The name of the library in which the job description for the autostart job entry resides.

Autostart job description name. The name of the job description for the autostart job entry about which information is being returned.

Autostart job name. The simple name of the job that is automatically started when the associated subsystem is started.

Communication entry job description library name. The name of the library in which the communications entry job description resides.

Communication entry job description name. The name of the job description used when a job is started as a result of receiving a program start request and processed through this communications entry. Possible special values follow:

USRPRF   The job description name that is specified in the user profile of the user that made the program start request is used for jobs that are processed through this communications entry.

Compare start position. The starting position for the routing data comparison. The comparison between the compare value and the routing data begins with this position in the routing data character string, and the last character position compared must be less than or equal to the length of the routing data used in the comparison.

Compare value. A value that is compared with the routing data to determine whether this is the routing entry that is used for starting a routing step. The special value *ANY means that any routing data is considered a match.

Control job (allocation). How the workstations that are associated with this job entry are allocated. Possible special values follow:

SIGNON   The workstations are allocated when the subsystem is started if the workstation is not already in use (signed on) in another subsystem. A sign-on prompt is displayed at each workstation that is associated with this work entry. If a workstation becomes allocated to a different subsystem, interactive jobs that are associated with the workstation are allowed to enter this subsystem through the Transfer Job (TFRJOB) command.

ENTER   The workstations that are associated with this work entry are not allocated when the subsystem is started. However, the interactive jobs that are associated with the workstations are allowed to enter this subsystem through the Transfer Job (TFRJOB) command.

Default user. The name of the default user profile used for evoke requests that enter the subsystem through this entry and contain no security information. Possible special values follow:

NONE   No user profile is specified as the default.

SYS   All user program start requests are treated the same as *NONE. For program start requests that are sent by system functions, the request runs under a predetermined user profile if a user profile is not specified on the program start request.

Device. The name of the device description or the type of the device being used with this communications entry. Possible special values follow:

ALL   All communications device types are used with this communications entry.

APPC   All advanced program-to-program communications devices can be used with this communications entry.

ASYNC   All asynchronous communications devices can be used with this communications entry.

BSCEL   All bisynchronous equivalency link communications devices can be used with this communications entry.

FINANCE   All finance communications devices can be used with this communications entry.

INTRA   All intrasystem communications devices can be used with this communications entry.

RETAIL   All retail communications devices can be used with this communications entry.

SNUF   All SNA upline facility communications devices can be used with this communications entry.

First class library name. The name of the library in which the first class resides.

First class name. The name of one of the two classes that the prestart jobs run under. Jobs start by using the first class that is specified until the number of jobs specified for the first class is reached. After the number of jobs that are specified for the first class is reached, then jobs are started by using the second class.

Format name specified. The format name as specified in the call to the API.

Initial number of jobs. The initial number of prestart jobs that are started when the subsystem is started.

Job description library name for type. The name of the library in which the job description for this workstation type resides.

Job description library name for workstation. The name of the library in which the job description for this workstation name resides.

Job description name for type. The name of the job description that is used for jobs started through this type of workstation entry. The possible special value follows:

USRPRF   The job description named in the user profile of the user that signs on at this type of workstation is used for jobs started through this entry.

Job description name for workstation. The name of the job description that is used for jobs started through this workstation name entry. The possible special value follows:

USRPRF   The job description named in the user profile of the user that signs on at this workstation is used for jobs started through this entry.

Maximum active jobs. The maximum number of jobs that can be active at the same time through this entry. If the entry specifies *NOMAX, indicating that there is no maximum, this number is -1.

Maximum active routing steps. The maximum number of routing steps (jobs) that can be active at the same time through this routing entry. If the routing entry specifies *NOMAX, indicating that there is no maximum, this number is -1.

Maximum number of jobs. The maximum number of prestart jobs that can be active at the same time for this prestart job entry. If the entry specifies *NOMAX, which indicates that there is no maximum, this number is -1.

Maximum number of uses. The maximum number of requests that can be handled by each prestart job in the pool before the job is ended. If the entry specifies *NOMAX, which indicates that there is no maximum, this number is -1.

Mode. The mode name of the communications device. The possible special value follows:

ANY   Any available modes defined to the communications device are allocated to the subsystem. If the communications device does not have defined modes associated with it, the communications device itself is allocated to the subsystem.

Number of jobs to use first class. The maximum number of jobs to run by using the first class. Possible special values follow:

-3   The system calculates how many prestart jobs use this class.

      If only one class is specified and -3 is specified, all of the jobs use that class.

      If two classes are specified and -3 is specified for both, the first class is the value of the maximum number of jobs field divided by two, and the second class is the value of the maximum number of jobs field minus the value that is calculated for the first class.

      If a specific number of jobs is specified for either class and -3 is specified for the other class, the system calculates the difference between maximum number of jobs and the specific number of jobs for the -3 designation.

-4   All of the prestart jobs use the specified class.

Number of jobs to use second class. The maximum number of jobs to run by using the second class. Possible special values follow:

-3   The system calculates how many prestart jobs use this class.

If only one class is specified and -3 is specified, all of the jobs use that class.

If two classes are specified and -3 is specified for both, the first class is the value of the maximum number of jobs field divided by two. The second class is the value of the maximum number of jobs field minus the value that is calculated for the first class.

If a specific number of jobs is specified for either class and -3 is specified for the other class, the system calculates the difference between maximum number of jobs and the specific number of jobs for the -3 designation.

-4   All of the prestart jobs use the specified class.

Pool identifier. The name of the subsystem pool identifier in which the prestart jobs will run.

Prestart job description library name. The name of the library in which the job description for the prestart job entry resides.

Prestart job description name. The name of the job description that is used for the prestart job entry. The possible special value follows:

*USRPRF   The job description that has the same name as the user profile that is used.

Prestart job name. The name of the prestart job.

Prestart job program library name. The name of the library in which the prestart job program resides.

Prestart job program name. The program name that is used to match an incoming request with an available prestart job.

Remote location. The name of the remote location for this entry.

Remote location entry job description library name. The name of the library in which the job description resides.

Remote location entry job description name. The name of the job description used when a job is started as a result of receiving a program start request and processed through this remote location entry. Possible special values follow:

*USRPRF   The job description name that is specified in the user profile of the user that made the program start request is used for jobs that are processed through this remote location entry.

Reserved. An ignored field.

Routing entry class library name. The name of the library in which the routing entry class resides.

Routing entry class name. The name of the class that is used when a routing step is started through this routing entry.

Routing entry pool identifier. The pool identifier of the storage pool in which the routing entry program is run.

Routing entry program library name. The name of the library in which the routing entry program resides.

Routing entry program name. The name of the program that is started when a routing step is started

through this routing entry in the subsystem description. If *RTGDTA is returned, the program name is taken from the routing data that was supplied and matched against this entry. The qualified program name will be taken from the routing data in this case, where the program name is specified in positions 37 through 46 and the library name is taken from positions 47 through 56.

Routing entry sequence number. The sequence number of the routing entry.

Second class library name. The name of the library in which the second class resides.

Second class name. One of the two classes that the prestart jobs run under. Jobs start by using the first class that is specified until the number of jobs specified for the first class is reached. After the number of jobs that are specified for the first class is reached, then jobs are started using the second class. The possible special value follows:

NONE   Only one class is used.

Start jobs. Whether the prestart jobs are started at the time the subsystem is started. Possible special values follow:

1   The prestart jobs are started at the time the subsystem is started.

0   The prestart jobs are not started at the time the subsystem is started. The Start Prestart Jobs (STRPJ) command is used to start these prestart jobs.

Subsystem library name specified. The name or special value specified in the call to this API for the library in which the subsystem description resides.

Subsystem library name used. The name of the library in which the subsystem description resides.

Subsystem name specified. The name of the subsystem specified in the call to this API.

Subsystem name used. The name of the subsystem about which information is being returned.

Threshold. The number at which additional prestart jobs are started. When the pool of available jobs (jobs available to service a program start request) is reduced below this number, more jobs (specified on the additional number of jobs field) are started and added to the available pool. This number is checked after a prestart job is attached to a procedure start request.

User profile name. The name of the user profile under which the prestart job runs.

User space library name specified. The name specified for the library that contains the user space to receive the generated list.

User space name specified. The name specified for the user space that is to receive the generated list.

Wait for job. Whether requests wait for a prestart job to become available or are rejected if a prestart job is not immediately available when the request is received. Possible special values follow:

1   Requests wait until there is an available prestart job, or until a prestart job is started, to handle the request.

0   Requests are rejected if a prestart job is not immediately available when the request is received.

Workstation name. The name of the workstation that is used by the subsystem. A generic workstation entry like DSP* is allowed.

Workstation type. The display device type. (See the TYPE keyword in the Communications Configuration

 book on the V5R1 Supplemental Manuals Web site.

Possible special values follow:

ALL — All workstation devices. This includes devices with 5250, ASCII, and 327x device types.

ASCII — ASCII display station.

NONASCII — All workstation devices that use a 5250 data stream. This includes the 327x device types.

CONS — System console display. This entry overrides a device type entry that specifies the same device type as the device being used as the console.

Error Messages

Message ID	Error Message Text
CPF1605 E	Cannot allocate subsystem description &1.
CPF1606 E	Error during allocation of subsystem &1.
CPF1607 E	Previous request pending for subsystem &1.
CPF1608 E	Subsystem description &1 not found.
CPF1619 E	Subsystem description &1 in library &2 damaged.
CPF1835 E	Not authorized to subsystem description.
CPF3CF1 E	Error code parameter not valid.
» CPF3CF2 E	Error(s) occurred during running of &1 API. «
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF811A E	User space &4 in &9 damaged.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.

CPF9838 E   User profile storage limit exceeded.

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

API introduced: V3R7

List Subsystem Job Queues (QWDLSJBQ) API

<div style="border:1px solid">

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	List format	Input	Char(8)
3	Qualified subsystem name	Input	Char(20)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

</div>

The List Subsystem Job Queues (QWDLSJBQ) API lists the job queues for a subsystem. It also gives the job queue allocation status, indicating whether the specified subsystem is active and has allocated this job queue or not. QWDLSJBQ replaces any data that already exists in the user space.

Authorities and Locks

User Space Authority
> *CHANGE

User Space Library Authority
> *EXECUTE

User Space Lock
> *EXCLRD

Subsystem Description Authority
> *USE

Subsystem Description Library Authority
> ≫*EXECUTE≪

Required Parameter Group

Qualified user space name
> INPUT; CHAR(20)
>
> The user space that receives the list, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name. You can use these special values for the library name:
>
> *CURLIB The job's current library
>
> *LIBL The library list

List format

INPUT; CHAR(8)

The format to use for the list of job queues. You can use the following format name:

SJQL0100   Basic job queue list. For details, see [Format of the Generated List](#).

Qualified subsystem name

INPUT; CHAR(20)

The subsystem about which to retrieve information, and the library in which the subsystem description is located. The first 10 characters contain the subsystem name, and the second 10 characters contain the library name. You can use these special values for the library name:

CURLIB   The job's current library

LIBL   The library list

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Format of the Generated List

The list of job queues that the QWDLSJBQ API returns into the user space consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section

The user area and generic header are described in [User Space Format for List APIs](#). For detailed descriptions of the fields in the tables, see [Field Descriptions](#).

Input Parameter Section

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

28	1C	CHAR(10)	Subsystem name
38	26	CHAR(10)	Subsystem library name specified

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Subsystem name used
10	A	CHAR(10)	Subsystem library name used

SJQL0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job queue name
10	A	CHAR(10)	Job queue library name
20	14	BINARY(4)	Sequence number
24	18	CHAR(10)	Allocation indicator
34	22	CHAR(2)	Reserved
36	24	BINARY(4)	Maximum active
40	28	BINARY(4)	Maximum by priority 1
44	2C	BINARY(4)	Maximum by priority 2
48	30	BINARY(4)	Maximum by priority 3
52	34	BINARY(4)	Maximum by priority 4
56	38	BINARY(4)	Maximum by priority 5
60	3C	BINARY(4)	Maximum by priority 6
64	40	BINARY(4)	Maximum by priority 7
68	44	BINARY(4)	Maximum by priority 8
72	48	BINARY(4)	Maximum by priority 9

Field Descriptions

Allocation indicator. A value indicating whether or not the job queue is allocated to the specified subsystem. Valid values are:

NO   The subsystem has not allocated this job queue. Either this subsystem is inactive, or another subsystem has allocated the job queue.

YES   The subsystem is active and has allocated this job queue.

Format name specified. The format name as specified in the call to the API.

Job queue library name. The name of the library in which the specified job queue resides.

Job queue name. The name of a job queue specified in a subsystem description job queue entry.

Maximum active. The maximum number of jobs that can be active at the same time through this job queue entry.

Maximum by priority 1 through 9. The maximum number of jobs that can be active at the same time for each priority level (1 through 9). A -1 in this field indicates that the value is *NOMAX.

Reserved. An ignored field.

Sequence number. The job queue entry sequence number. The subsystem uses this number to determine the order in which job queues are processed. Jobs from the queue with the lowest sequence number are processed first.

Subsystem library name specified. The name or special value specified in the call to this API for the library in which the subsystem description resides.

Subsystem library name used. The name of the library in which the subsystem description resides.

Subsystem name. The name of the subsystem about which information is being returned.

Subsystem name used. The name of the subsystem about which information is being returned.

User space library name specified. The library name or special value specified in the call to this API.

User space name. The name of the user space that receives the list.

Error Messages

Message ID	Error Message Text
CPF1605 E	Cannot allocate subsystem description &1.
CPF1606 E	Error during allocation of subsystem &1.
CPF1607 E	Previous request pending for subsystem &1.
CPF1608 E	Subsystem description &1 not found.
CPF1619 E	Subsystem description &1 in library &2 damaged.
CPF1835 E	Not authorized to subsystem description.
CPF3CF1 E	Error code parameter not valid.
» CPF3CF2 E	Error(s) occurred during running of &1 API.«
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF811A E	User space &4 in &9 damaged.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.

CPF9810 E   Library &1 not found.
CPF9820 E   Not authorized to use library &1.
CPF9830 E   Cannot assign library &1.
CPF9838 E   User profile storage limit exceeded.
CPF9872 E   Program or service program &1 in library &2 ended. Reason code &3.

Move Job (QSPMOVJB) API

```
Required Parameter Group:

    1      Function information              Input        Char(*)
    2      Length of function information    Input        Binary(4)
    3      Function information format       Input        Char(8)
    4      Error code                        I/O          Char(*)


Default Public Authority: *USE

Threadsafe: No
```

The Move Job (QSPMOVJB) API performs one of two functions. They are to:

- Move one job at a time to the top of a job queue
- Move a job after another job

The jobs are identified by the internal job identifier or by the qualified job name. See How to Specify Job Identifying Fields for the valid combinations of identifying jobs.

As a result of using this API, a job could:

- Change job queues

  The job changes job queues when it is moved to the top of another job queue.

  The job being moved resides on the job queue of the target job.

- Change priority

  The job priority is changed to the requester's highest schedule priority if the job is moved to the top of a job queue.

```
 Requester's priority = 2


                              Job            Priority
 New priority = 2 when moved here-------------->
                              A                 3
                              B                 5
```

  The job priority will be changed to the priority of the target job if the requester's highest schedule priority is equal to or less than the target job.

```
 Requester's priority = 2


                              Job            Priority


                              C                 3
 New priority = 3 when moved here------------->
                              D                 5
```

The job priority is changed to the requester's highest schedule priority if:

❍ The requester's priority is greater than the target job and

❍ The requester's priority is greater or equal to the jobs it is moving ahead of.

```
Requester's priority = 4

                            Job           Priority

                             C               3
New priority = 4 when moved here------------->
                             D               5


Requester's priority = 4

                            Job           Priority

                             E               3
New priority = 4 when moved here------------->
                             F               4

The owner of the job moving must also be
the owner of job F.
```

- Be held

  A job in ready status is held when it is moved after a job that is held.

- Be released

  A job in held status is released when it is moved to the top of a job queue.

  A job in held status is released when it is moved after a job that is released.

Restrictions for Movement of Jobs

The Move Job API has restrictions that determine whether a job can be moved.

- Scheduled jobs cannot be referenced as the source job or the target job.

- The job to be moved must exist, be a batch job, and be on a job queue.

- The job queue must exist.

- If a job is held with hold spooled files specified, and the job is to be moved to the top of a queue, no spooled file must exist for the job.

Authorities and Locks

Job Authorities: The requester is authorized to the job if one or more of the following conditions are met.

- The requester is the owner of the job to be moved.

- The requester has *JOBCTL authority.

Job Queue Authority

Authority to the target job queue and target job queue library

      *USE

Job queue lock on which the source job resides

      *EXCLRD

Job queue lock on which the target job resides

      *EXCLRD

User Profile Highest Schedule Priority

- The requester must have a priority limit of at least x-1 (x being the priority of the job that will follow after this job).
- If the requester has equal priority to a job that it is moving ahead of, the requester must own all of the jobs it is moving ahead of at that priority level or have *JOBCTL authority.

Required Parameter Group

Function information

      INPUT; CHAR(*)

      The information that is associated with the job or jobs to be moved and the job queue to which the jobs are to be moved. See the [Format of the Function Information](#) for the format of this parameter.

Length of function information

      INPUT; BINARY(4)

      The length of the function information in the function information parameter. The length depends on the function format. Each format has a different (but fixed) length as shown in the specific format tables. The minimum length for format MJOB0100 is 62 bytes; the minimum length for format MJOB0200 is 84 bytes.

Function information format

      INPUT; CHAR(8)

      The format of the function information that is being provided. The information is provided in the function information parameter. They are:

    MJOB0100   Format for the function of *NEXT, which is moving one job at a time to the top of a job queue.

    MJOB0200   Format for the function of *AFTER, which is moving a job after another job.

Error code

      I/O; CHAR(*)

      The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Format of the Function Information

MJOB0100 Format: The following table shows the information for the MJOB0100 format. For more details about the fields in the following tables see, Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Source job name
10	A	CHAR(10)	Source job user name
20	14	CHAR(6)	Source job number
26	18	CHAR(16)	Source internal job identifier
42	2A	CHAR(10)	Target job queue name
52	34	CHAR(10)	Target job queue library name

MJOB0200 Format: The following table shows the information for the MJOB0200 format. For more details about the fields in the following tables see, Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Source job name
10	A	CHAR(10)	Source job user name
20	14	CHAR(6)	Source job number
26	18	CHAR(16)	Source internal job identifier
42	2A	CHAR(10)	Target job name
52	34	CHAR(10)	Target job user name
62	3E	CHAR(6)	Target job number
68	44	CHAR(16)	Target internal job identifier

Field Descriptions

Source internal job identifier. The internal identifier for the job to be moved. The identifier is not valid following an initial program load (IPL). If you attempt to use it after an IPL, an exception occurs. This field must be blank when a source job name is given. Use one of the following APIs to make the identifier available:

> List Job (QUSLJOB) API

> Retrieve Job Information (QUSRJOBI) API

There may be a performance advantage when identifying the source job by its internal identifier.

Source job name. The name of the job to be moved. The possible values are:

INT The job to be moved is identified by the internal job identifier.

Name   The name of the job to be moved.

Source job number. The number of the job to be moved. It can optionally be blank when a source job name or a source job user name or both are specified. This field must be blank when the source job name is specified as *INT.

Source job user name. The user name of the job to be moved. It can optionally be blank when a source job name is specified. This field must be blank when the source job name is specified as *INT.

Target internal job identifier. The internal identifier for the job after which the source job is to be moved. The identifier is not valid following an initial program load (IPL). If you attempt to use it after an IPL, an exception occurs. This field must be blank when a target job name is given. Use one of the following APIs to make the identifier available:

List Job (QUSLJOB) API

Retrieve Job Information (QUSRJOBI) API

There may be a performance advantage when identifying the target job by its internal identifier.

Target job name. The name of the job after which the source job is to be moved. The possible values are:

INT   The target job is identified by the internal job identifier.

Name   The name of the job after which the source job is to be moved.

Target job number. The number of the target job. It can optionally be blank when a target job name or target job user name or both are specified. This field must be blank when the target job name is specified as *INT.

Target job queue library name. The name of the library that contains the job queue. This must be specified when a target job queue name is given. The possible values are:

LIBL   The library list is used to locate the job queue.

CURLIB   The current library is used to locate the job queue. If no library is specified as the current library for the job, QGPL is used.

Name   The library name.

Blanks   No library name is given when the job queue name is *SAME.

Target job queue name. The name of the job queue to which the job is to move. The possible values are:

SAME   The job will move to the top of the job queue on which it currently resides.

Name   The name of the job queue to which the job is to move to the top of.

Target job user name. The user name of the target job. It can optionally be blank when a target job name is specified. This field must be blank when the target job name is specified as *INT.

How to Specify Job Identifying Fields

This table illustrates the valid combinations of values for format MJOB0100.

Qualified Job Name			Internal Job Identifier	Job Queue
Job Name	**User Name**	**Job Number**		
Name	Name	Number	Blanks	Name
Name	Name	Blanks	Blanks	Name
Name	Blanks	Number	Blanks	Name
Name	Blanks	Blanks	Blanks	Name
*INT	Blanks	Blanks	Internal job identifier	Name

This table illustrates the valid combinations of values for format MJOB0200.

The qualified job name can use the same combinations of the qualified job name specified in format MJOB0100.

Source Qualified Job Name	Source Internal Job Identifier	Target Qualified Job Name	Target Internal Job Identifier
Qualified job name	Blanks	Qualified job name	Blanks
Qualified job name	Blanks	*INT	Internal Job identifier
*INT	Internal Job identifier	Qualified job name	Blanks
*INT	Internal Job identifier	*INT	Internal job identifier

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF33C1 E	Job &3/&2/&1 not on job queue. Job not moved.
CPF3CF1 E	Error code parameter not valid.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C42 E	User name or job number is not blank.
CPF3C43 E	Internal job identifier is not valid.
CPF3C52 E	Internal job identifier no longer valid.
CPF3C53 E	Job &3/&2/&1 not found.
CPF3C58 E	Job name specified is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF33BA E	Job queue &1 in library &2 is not valid.
CPF33B3 E	Not authorized to job queue &1.
CPF33B4 E	Not authorized to job &3/&2/&1. Job not moved.
CPF33B5 E	Job &3/&2/&1 is not available for moving.
CPF33B6 E	Job &3/&2/&1 held by HLDJOB command. Job not moved.

CPF33B7 E	Job &3/&2/&1 specified more than once. Job not moved.
CPF33B8 E	Priority required to move job &3/&2/&1 exceeds limit of user &9.
CPF33B9 E	Priority required to move job &3/&2/&1 exceeds limit of user &9.
CPF3330 E	Necessary resource not available.
CPF3343 E	Duplicate job names found.
CPF8121 E	&8 damage on job queue &4 in library &9.
CPF8122 E	&8 damage on library &4.
CPF9801 E	Object &2 in library &3 not found.
CPF9810 E	Library &1 not found.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

Open List of Activation Attributes (QWVOLACT) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	List information	Output	Char(80)
4	Number of records to return	Input	Binary(4)
5	Format name	Input	Char(8)
6	Activation group number	Input	Binary(4)
7	Qualified job name	Input	Char(26)
8	Internal job identifier	Input	Char(16)
9	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Open List of Activation Attributes (QWVOLACT) API generates a list of all the activation attributes that are associated with an activation group in a given job. The QWVOLACT API places the list into a receiver variable. You can access additional records by using the Get List Entries (QGYGTLE) API. On successful completion of the QWVOLACT API, a handle is returned in the list information parameter. You may use this handle on subsequent calls to the following APIs:

Get List Entries (QGYGTLE)

Find Entry Number in List (QGYFNDE)

Close List (QGYCLST)

The records returned by QWVOLACT include an information status field that describes the completeness and validity of the information. Be sure to check the information status field before using any other information returned.

Authorities and Locks

Job Authority

*JOBCTL if the job for which activation group attributes are being retrieved has a user profile different from that of the job that calls the QWVOLACT API.

For additional information on these authorities, see the [iSeries Security Reference](#) book.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is used to return the activation attributes that you requested.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

List information

OUTPUT; CHAR(80)

Information about the list of activation attributes that were opened. For a description of the layout of this parameter, see .

Number of records to return

INPUT; BINARY(4)

The number of records in the list to put into the receiver variable.

Format name

INPUT; CHAR(8)

The format of the information to be returned in the receiver variable. You must use the following format name:

RACT0100   This format is described in [RACT0100 Format](#).

Activation group number

INPUT; BINARY(4)

The number of the activation group that will be used to locate the activations whose attributes are to be returned. You can use this special value for the activation group number:

-1   The activation attributes for all activation groups are returned.

Qualified job name

INPUT; CHAR(26)

The job name, the job user profile, and the job number of the job for which you want to return activation attributes.

CHAR 1-10   The job name

CHAR 11-20   The user profile

CHAR 21-26   The job number

You can use these special values for the qualified job name:

		The job in which this program is running. The rest of the qualified job name parameter must be blank.

    *   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 rest of the qualified job name parameter must be blank.

Internal job identifier

    INPUT; CHAR(16)

    The internal name for the job. The List Job (QUSLJOB) API creates this identifier. If you do not specify *INT for the qualified job name parameter, this parameter must contain blanks.

    If your application already has this information available from the QUSLJOB API, the QWVOLACT API can locate the job more quickly with this information than with a job name. However, if you call QUSLJOB solely to obtain this parameter for use by QWVOLACT, you would get poorer performance than by using a job name in calling QWVOLACT.

Error code

    I/O; CHAR(*)

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

RACT0100 Format

The following table shows the information returned in each record in the receiver variable for the RACT0100 format. For a detailed description of each field, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Activation group name
10	A	CHAR(6)	Reserved
16	10	BINARY(4)	Activation group number
20	14	BINARY(4)	Reserved
24	18	BINARY(4)	Activation number
28	1C	BINARY(4)	Static storage size
32	20	CHAR(10)	Program name
42	2A	CHAR(10)	Program library
52	34	CHAR(1)	Program type
53	35	CHAR(11)	Reserved

Field Descriptions

Activation number. The activation number of the activation listed. This is an internal number that uniquely identifies the activation within the job.

Activation group name. The name of the activation group that contains the attributes listed. Possible values follow:

DFTACTGRP   The activation group is one of the default activation groups.

UNNAMED   The activation group does not have a name.

Activation group number. The activation group number of the activation group listed. This is an internal number that uniquely identifies the activation group within the job.

Program library. The name of the library that contains the program that this activation is for. Possible values follow:

N   The program no longer exists in the system.

Program name. The name of the program that this activation is for. Possible values follow:

N   The program no longer exists in the system.

Program type. The type of call that this activation is for. Possible values follow:

0   The type was a program or *PGM.

1   The type was a service program or *SRVPGM.

Reserved. An ignored field.

Static storage size. The total amount of static storage allocated to the activation in bytes.

Error Messages

Message ID	Error Message Text
CPF0941 E	Job &3/&2/&1 no longer in system.
CPF1071 E	No authority to job &3/&2/&1.
CPF136A E	Job &3/&2/&1 not active.
CPF136B E	Job &3/&2/&1 in use.
CPF136C E	Value &1 for activation group number not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF2401 E	Not authorized to library &1.
CPF3C19 E	Error occurred with receiver variable specified.

CPF3C21 E	Format name &1 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.
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.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0027 E	&1 is not valid for number of records to return.

API introduced: V4R2

Open List of Activation Group Attributes (QWVOLAGP) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	List information	Output	Char(80)
4	Number of records to return	Input	Binary(4)
5	Format name	Input	Char(8)
6	Qualified job name	Input	Char(26)
7	Internal job identifier	Input	Char(16)
8	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Open List of Activation Group Attributes (QWVOLAGP) API generates a list of all the activation groups that are associated with a given job and their attributes. The QWVOLAGP API places the list into a receiver variable. You can access additional records by using the Get List Entries (QGYGTLE) API. On successful completion of the QWVOLAGP API, a handle is returned in the list information parameter. You may use this handle on subsequent calls to the following APIs:

> Get List Entries (QGYGTLE)

> Find Entry Number in List (QGYFNDE)

> Close List (QGYCLST)

The records returned by the QWVOLAGP API include an information status field that describes the completeness and validity of the information. Be sure to check the information status field before using any other information returned.

Authorities and Locks

Job Authority

> ❍ *JOBCTL if the job for which activation group attributes are being retrieved has a user profile different from that of the job that calls the QWVOLAGP API.

> For additional information on these authorities, see the [iSeries Security Reference](#) book.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is used to return the activation group attributes that was requested.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable.

List information

> OUTPUT; CHAR(80)

> Information about the list of activation group attributes that were opened. For a description of the layout of this parameter, see Format of Open List Information.

Number of records to return

> INPUT; BINARY(4)

> The number of records in the list to put into the receiver variable.

Format name

> INPUT; CHAR(8)

> The format of the information to be returned in the receiver variable. You must use the following format name:

> *RAGA0100*   This format is described in RAGA0100 Format.

Qualified job name

> INPUT; CHAR(26)

> The job name, the job user profile, and the job number of the job for which you want to return activation group attributes.

> *CHAR 1-10*   The job name

> *CHAR 11-20*   The user profile

> *CHAR 21-26*   The job number

> You can use these special values for the qualified job name:

> *   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 rest of the qualified job name parameter must be blank.

Internal job identifier

> INPUT; CHAR(16)

The internal name for the job. The List Job (QUSLJOB) API creates this identifier. If you do not specify *INT for the qualified job name parameter, this parameter must contain blanks.

If your application already has this information available from the List Job (QUSLJOB) API, the QWVOLAGP API can locate the job more quickly with this information than with a job name. However, if you call QUSLJOB solely to obtain this parameter for use by QWVOLAGP, you would get poorer performance than by using a job name in calling QWVOLAGP.

Error code

I/O; CHAR(*)

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

RAGA0100 Format

The following table shows the information returned in the list data section of the receiver variable for the RAGA0100 format. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Activation group name
10	A	CHAR(6)	Reserved
16	10	BINARY(4)	Activation group number
20	14	BINARY(4)	Number of activations
24	18	BINARY(4)	Number of heaps
28	1C	BINARY(4)	Static storage size
32	20	BINARY(4)	Heap storage size
36	24	CHAR(10)	Root program name
46	2E	CHAR(10)	Root program library
56	38	CHAR(1)	Root program type
57	39	CHAR(1)	Activation group state
58	3A	CHAR(1)	Shared activation group indication
59	3B	CHAR(1)	In-use indicator
60	3C	CHAR(4)	Reserved

Field Descriptions

Activation group name. The name of the activation group that contains the attributes listed. Possible values follow:

DFTACTGRP   The activation group is one of the default activation groups.

UNNAMED   The activation group does not have a name.

Activation group number. The activation group number of the activation group listed. This is an internal number that uniquely identifies the activation group within the job.

Activation group state. The state of the activation group. Possible values follow:

0   The activation group is in user state.

1   The activation group is in system state.

Heap storage size. The total amount of heap storage that is allocated to the activation group in bytes.

In-use indicator. Whether the activation group is eligible to be reclaimed. An activation group can be reclaimed by the Reclaim Activation Group (RCLACTGRP) command. Possible values follow:

0   The activation group is not in use and is eligible to be reclaimed.

1   The activation group is in use and cannot be reclaimed.

Note: It is not recommended to reclaim eligible activation groups if you are not familiar with that activation group. Other activation groups may have references to the activation group that you are reclaiming. For example, a program of one activation group could be bound to a service program that belongs to another activation group. If you reclaim the service program's activation group and then call the program, you will get a destroyed object error when the service program is referred to.

Number of activations. The total number of program activations in this activation group.

Number of heaps. The total number of heaps that are allocated by this activation group.

Reserved. An ignored field.

Root program library. The name of the library that contains the program that caused this activation group to be created. The possible value follows:

*N   The program no longer exists in the system.

Note: When the activation group is the default activation group, there is no root program. Blanks are returned in this case.

Root program name. The name of the program that caused this activation group to be created. The possible value follows:

*N   The program no longer exists in the system.

Note: When the activation group is the default activation group, there is no root program. Blanks are returned in this case.

Root program type. The type of program that caused this activation group to be created. Possible values follow:

N   The program no longer exists in the system.

0   The type is a program or *PGM.

1   The type is a service program or *SRVPGM.

2   The type is a Java program.

Note: When the activation group is the default activation group, there is no root program. A blank is returned in this case.

Shared activation group indication. Whether the activation group is shared or not. A shared activation group is an activation group that belongs to more than one job at the same time. Possible values follow:

0   The activation group is not shared with other jobs.

1   The activation group is shared with other jobs.

Static storage size. The total amount of static storage allocated to the activation group in bytes.

Error Messages

Message ID	Error Message Text
CPF0941 E	Job &3/&2/&1 no longer in system.
CPF1071 E	No authority to job &3/&2/&1.
CPF136A E	Job &3/&2/&1 not active.
CPF136B E	Job &3/&2/&1 in use.
CPF136C E	Value &1 for activation group number not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF2401 E	Not authorized to library &1.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 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.
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.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0027 E	&1 is not valid for number of records to return.

API introduced: V4R2

Open List of Job Queues (QSPOLJBQ) API

Required Parameter Group:

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

Default Public Authority: *USE Threadsafe: Yes

The Open List of Job Queues (QSPOLJBQ) API generates a list of job queues on the system. The list can include all job queues on the system, all job queues for a specified library list, the allocated job queues for a specified active subsystem, or the defined job queues for a specified active subsystem (this would include both the allocated job queues and the job queues the subsystem could not allocate), or all the allocated job queues for all the active subsystems. The filtered list can then be sorted depending on the value of the sort parameter. When requesting job queue information for all job queues on the system or for a specific job queue within a library, if the signed-on user is not authorized to the library of the job queue, information for that job queue in not returned by this API. When requesting the job queue information for job queues defined to an active subsystem, all job queue information will be returned with out regard to the users authority to the subsystem. Upon successful completion of this API, a handle is returned in the list information parameter. You may use this handle on subsequent calls to the following APIs:

> Get List Entries (QGYGTLE)

> Find List Entry (QGYFNDE)

> Close List (QGYCLST)

Performance Impacts

Sorting on one or more values of job queue name, job queue library name, job queue status, subsystem name, subsystem library name, number of jobs on job queue, sequence number, maximum active, current active or description takes more processing power and time.

Authorities and Locks

Job Queue Library Authority

> *EXECUTE

Job Queue Lock

> This API gets an *EXCLRD lock on the job queue.

Subsystem Description Lock

> This API gets an *EXCLRD lock on the subsystem description.

This API does not check the caller's authority to the subsystem description or subsystem description library when retrieving the subsystem description information.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable used to return the job queue information.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

Format of receiver variable

INPUT; CHAR(8)

The format of the job queue information being returned. You can specify the following:

OJBQ0100 Contains the basic information about the job queue. For more information about the OJBQ0100 format, see Format of Receiver Variable.

List information

OUTPUT; CHAR(80)

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

Filter information

INPUT; CHAR(*)

The information in this parameter is used for filtering the list of job queue information. For more information about the filter information, see Filter Information.

Sort information

INPUT; CHAR(*)

Information on what fields within the record of information to sort. See Format of Sort Information for a description of the layout of this parameter. Note that when sorting is requested, the entire list has to be built and sorted before any records can be returned.

Number of records to return

INPUT; BINARY(4)

The number of records in the list to put into the receiver variable.

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

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

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

Error code

I/O; CHAR(*)

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

Filter Information

The following table shows the format of the filter information parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of filter information
4	4	CHAR(10)	Job queue name
14	E	CHAR(10)	Job queue library name
24	18	CHAR(10)	Active subsystem name
34	22	CHAR(*)	Reserved

Format of Sort Information

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

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of keys to sort on
Offsets vary. These fields repeat for each sort key field.		BINARY(4)	Sort key field starting position
		BINARY(4)	Sort key field length
		BINARY(2)	Sort key field data type
		CHAR(1)	Sort order
		CHAR(1)	Reserved

Note: If the last three fields (sort key field data type, sort order, and the reserved field) are not used, then they must be set to hexadecimal zeros. This causes all the data to be treated as character data, and it is sorted in ascending order.

Field Descriptions

Active subsystem name. The active subsystem name whose job queue information is to be returned. A simple active subsystem name or one of the following special values may be specified.

simple name   A simple name of an active subsystem. The job queue name field must be set to special value *ALLOCATED or *DEFINED.

ALL	All allocated job queues for all active subsystems on the system are returned. The Job queue name field must be set to the special value *ALLOCATED.
blanks	The active subsystem name field is ignored. Only the job queue name field is used.

If the active subsystem field is used, the job queue name field must be set to the correct special value and the job queue library name field must be set to blanks.

Job queue name. The job queue about which to retrieve information. A simple job queue name, a generic job queue name, or a special value may be specified.

The following values require the active subsystem name field to be blank.

simple name	A simple name of a job queue.
generic name	A generic name for job queues.
ALL	All job queues are returned.

When the active subsystem name field is set to a simple subsystem name, then one of the following special values must be used:

ALLOCATED	Only the job queues that have been allocated by the active subsystem are returned.
DEFINED	The job queues that are defined to the active subsystem. This includes the job queues that have been allocated by the subsystem.

When the active subsystem name field is set to *ALL, the following special value must be used:

ALLOCATED	Only the job queues that have been allocated by the active subsystem name are returned.

Job queue library name. The library in which the job queue is located. A specific library name or a special value may be specified.

The following values apply only when the job queue name field is set to a simple job queue name, a generic job queue name, or the special value *ALL.

simple name	A simple name of a library.
ALL	All libraries on the system are searched.
ALLUSR	All user defined libraries to which the user is authorized are searched.
CURLIB	The job's current library is searched.
LIBL	The library list for the job is searched.
USRLIBL	The libraries in the user portion of the job's library list are searched.

When the job queue name field is set to *ALLOCATED or *DEFINED the the job queue library field must be blanks.

Length of filter information. The length of the filter information. An error message is returned if the length of filter information is not set correctly.

Number of keys to sort on. The number of fields within the record structure on which to sort. If 0 is specified, the list is not sorted.

Reserved. Must be set to hexadecimal zeros.

Sort key field data type. Data type of field to sort. See the [Sort](#) (QLGSORT) API for information on the list of data types available.

Sort key field length. The length of the field on which to sort.

Sort key field starting position. Within the record of information, the starting position of the field on which to sort.

Sort order. Whether the list should be sorted in ascending or descending order according to the key. See the [Sort](#) (QLGSORT) API for information on the sort order special values.

Format of Receiver Variable

The following tables describe the order and format of the data returned in the receiver variable.

OJBQ0100 Format

For more details about the fields in the following table, see [Field Descriptions](#).

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job queue name
10	A	CHAR(10)	Job queue library name
20	14	CHAR(1)	Job queue status
21	15	CHAR(10)	Subsystem name
31	15	CHAR(10)	Subsystem library name
41	29	CHAR(3)	Reserved
44	2C	BINARY(4)	Number of jobs on job queue
48	30	BINARY(4)	Sequence number
52	34	BINARY(4)	Maximum active
56	38	BINARY(4)	Current active
60	3C	CHAR(50)	Description

Field Descriptions

Current active. The number of jobs currently running in the active subsystem from this job queue. This field is -1 if the job queue is not allocated, is damaged, does not exist, or the job queue has not been allocated by the subsystem that was specified in the active subsystem field in the filter parameter.

Description. The text description for this job queue. This field will be blank if the job queue is defined to an active subsystem, but has not been created or the job queue is damaged.

Job queue library name. The name of the library in which the job queue is located.

Job queue name. The name of the job queue.

Job queue status. The current status of the job queue. The possible values are:

0   The job queue is currently held. No jobs can become active from this job queue.

1   The job queue is released. Jobs can become active from this queue.

2   The job queue is damaged.

3   The job queue is defined to the active subsystem, but has not been created. No jobs can become active from this job queue until it is created.

Maximum active. The maximum number of jobs that can be active in the subsystem from this job queue at one time. A -1 in this field indicates that the value is *NOMAX. This field is -2 if the job queue has not been defined to an active subsystem or the job queue is damaged.

Number of jobs on job queue. The total number of jobs currently waiting to run on this job queue. This field ise -1 if the job queue is defined to the active subsystem, but has not been created or the job queue is damaged.

Reserved. A reserved field.

Sequence number. The job queue entry sequence number. The subsystem uses this number to determine the order in which the job queues are processed. Jobs from the job queue with the lowest sequence number in the job queue are selected first. This field is -1 if the job queue has not been defined to an active subsystem or the job queue is damaged.

Subsystem name. The name of the subsystem to which this job queue is allocated. If the job queue has been allocated by a different subsystem than was specified in the filter parameter, the subsystem name will identify the subsystem to which the job queue is allocated. This field is blank if the job queue is not allocated, is damaged, or does not exist.

Subsystem library name. The library in which the subsystem description resides. This field will be blank if the job queue is not allocated, damaged or does not exist.

Error Messages

Message ID	Error Message Text
CPF1054 E	No subsystem &1 active.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.

CPF9804 E	Object &2 in library &3 damaged.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9871 E	Error occurred while processing.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0024 E	&1 is not valid for number of keys to sort on.
GUI0025 E	&1 is not valid for sort key field starting position.
GUI0026 E	&1 is not valid for sort key field length.
GUI0027 E	&1 is not valid for number of records to return.
GUI0119 E	Reserved field in sort information not valid.
GUI0120 E	Order field in sort information not valid.
GUI0150 E	Subsystem key must be specified.
GUI0152 E	&1 is not valid for length of filter information.

API introduced: V5R1

Open List of Jobs (QGYOLJOB) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Receiver variable definition information	Output	Char(*)
5	Length of receiver variable definition information	Input	Binary(4)
6	List Information	Output	Char(80)
7	Number of records to return	Input	Binary(4)
8	Sort information	Input	Char(*)
9	Job selection information	Input	Char(*)
10	Size of job selection information	Input	Binary(4)
11	Number of fields to return	Input	Binary(4)
12	Key of fields to be returned	Input	Array(*) of Binary(4)
13	Error Code	I/O	Char(*)

Optional Parameter Group 1:

14	Job selection format name	Input	Char(8)

Optional Parameter Group 2:

15	Reset status statistics	Input	Char(1)
16	General return data	Output	Char(*)
17	Length of general return data	Input	Binary(4)

Default Public Authority: *USE

Threadsafe: Conditional; see [Usage Notes](#).

The Open List of Jobs (QGYOLJOB) API generates a list of jobs on the system. The list is based on specified selection criteria. The filtered list may then be sorted depending on the value of the sort parameter. Upon successful completion of this API, a handle is returned in the list information parameter. You may use this handle on subsequent calls to the following APIs:

   Get List Entries (QGYGTLE)

   Find List Entry (QGYFNDE)

   Close List (QGYCLST)

Authorities and Locks

None.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is used to return the job information.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable. This must be large enough to hold at least one record of information.

Format name

> INPUT; CHAR(8)

> The format of the job list to be returned. If format OLJB0200 is specified, the fields that are selected by the caller are returned for each job in the list. This format is slower than the OLJB0100 format. The performance varies depending on the number of fields selected.

> The possible format names follow:

> > *OLJB0100*   Basic job list.

> > *OLJB0200*   Basic job list with keyed return fields.

> > *OLJB0300*   Active job list with keyed return fields.

> See [Format of Receiver Variable](#) for more information.

Receiver variable definition information

> OUTPUT; CHAR(*)

> The variable that is used to return the description of how the keyed portion of the list is returned in the receiver variable. This variable is not filled in if the OLJB0100 format is used. See [Format of Receiver Variable Definition Information](#) for a description of the layout of this parameter.

Length of receiver variable definition information

> INPUT; BINARY(4)

> The length of the receiver variable definition information. This must be large enough to hold the information for all key fields that are specified in the key of fields to be returned parameter.

List Information

> OUTPUT; CHAR(80)

> Information about the list that is created by this program. For a description of the layout of this parameter, see [Format of Open List Information](#).

Number of records to return

INPUT; BINARY(4)

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

Sort information

INPUT; CHAR(*)

Information on what fields within the record of information to sort. See [Format of Sort Information](#) for a description of the layout of this parameter.

Job selection information

INPUT; CHAR(*)

Information that is used for selecting which jobs to include in the list. See [Format of Job Selection Information](#) for a description on the layout of this parameter.

Size of job selection information

INPUT; BINARY(4)

The size, in bytes, of the job selection information parameter. The minimum value required for this parameter is 60 (or 108 with selection format OLJS0200).

Number of fields to return

INPUT; BINARY(4)

The number of fields to return. If this parameter is 0, the key of fields to be returned parameter is not used, the receiver variable definition information parameter is not modified, and the length of receiver variable definition information must be 0. This field must be set to 0 when the format is set to OLJB0100.

Key of fields to be returned

INPUT; ARRAY(*) of BINARY(4)

The list of the fields to be returned. For a list of the valid key fields, see [Valid Keys](#). For a list of valid key fields allowed on the OLJB0300 format, see [List of keys supported for format OLJB0300](#).

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

Job selection format name

INPUT; CHAR(8)

The format of the job selection information input. If this parameter is not specified, the OLJS0100 format is used. The possible format names follow:

OLJS0100   Basic selection information.

OLJS0200   Additional selection information.

See the [Format of Job Selection Information](#) for more information.

Optional Parameter Group 2

Optional Parameter Group 2 applies only when the format is OLJB0300.

Reset status statistics

INPUT; CHAR(1)

The elapsed time and all the key fields that are based on the elapsed time are reset to zero. If a format other than OLJB0300 is specified, this field needs to be zero. The default value for this field is zero. The following special values may be specified:

0 The elapsed time and the key fields based on the elapsed time are not reset.

1 The elapsed time and the key fields based on the elapsed time are reset back to zero.

General return data

OUTPUT; CHAR(*)

General output information that applies to the OLJB0300 format. See [General return data](#) for a description of the data returned. This field is used only if the Length of general return data is greater than 8.

Length of general return data

INPUT; BINARY(4)

The length of the general output information. The general output information field must be a minimum of 8 bytes or 0 when the format is not OLJB0300. The default value for this field is 0.

Format of Receiver Variable

The following tables describe the order and format of the data that is returned in the receiver variable.

OLJB0100 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name used
10	A	CHAR(10)	User name used
20	14	CHAR(6)	Job number used
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(10)	Status
52	34	CHAR(1)	Job type
53	35	CHAR(1)	Job subtype
54	36	CHAR(2)	Reserved

OLJB0200 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(56)	Everything in OLJB0100 format
56	38	CHAR(1)	Job information status
57	39	CHAR(3)	Reserved
This information repeats for each key selected.		CHAR(*) or BINARY(4)	Data

OLJB0300 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name used
10	A	CHAR(10)	User name used
20	14	CHAR(6)	Job number used
26	1A	CHAR(4)	Active job status
30	1E	CHAR(1)	Job type
31	1F	CHAR(1)	Job subtype
32	20	BINARY(4)	Total length of data returned
36	24	CHAR(4)	Reserved
This information repeats for each key selected.		CHAR(*) or BINARY(4)	Data

Field Descriptions

Active job status. The active status of the initial thread of the job. For the list of possible values, see the active job status field under Field Descriptions.

Data. The data returned for the key field.

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

Job information status. Whether the information was available for the job. The possible values follow:

blank   The information was available.

L   The information was not available because the job was not accessible.

Job name used. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the workstation where the job started. For a batch job, you specify the name in the command when you submit the job.

Job number used. The system-assigned job number.

Job subtype. Additional information about the job type (if any exists). Refer to Comparing Job Type and Subtype with the Work with Active Job Command in the Retrieve Job Information (QUSRJOBI) API for information about how the job type field and the job subtype field equate to the type field in the Work with Active Job (WRKACTJOB) command. The possible values follow:

blank   The job has no special subtype.

D   The job is a batch immediate job.

E   The job started with a procedure start request.

F   The job is an iSeries Advanced 36 machine server job.

J   The job is a prestart job.

P   The job is a print driver job.

T   The job is a System/36 multiple requester terminal (MRT) job.

U   The job is an alternate spool user.

Job type. The type of job. Refer to Comparing Job Type and Subtype with the Work with Active Job Command in the Retrieve Job Information (QUSRJOBI) API for information about how the job type field and the job subtype field equate to the type field in the Work with Active Job (WRKACTJOB) command. The possible values for this field follow:

A   The job is an autostart job.

B   The job is a batch job.

I   The job is an interactive job.

M   The job is a subsystem monitor job.

R   The job is a spooled reader job.

S   The job is a system job.

W   The job is a spooled writer job.

X   The job is the start-control program-function (SCPF) system job.

» *blank*   The job has no type. A possible reason is the job was not found.«

Reserved. An ignored field.

Status. The status of the job. The valid values follow:

ACTIVE   The job has started, and it can use system resources (processing unit, main storage, and so on). This value does not guarantee that the job is currently running, however. For example, an active job may be in one of the following states where it is not in a position to use system resources:

- The Hold Job (HLDJOB) command holds the job; the Release Job (RLSJOB) command allows the job to run again.
- The Transfer Group Job (TFRGRPJOB) or Transfer Secondary Job (TFRSECJOB) command suspends the job. When control returns to the job, the job can run again.
- The job is disconnected using the Disconnect Job (DSCJOB) command. When the interactive user signs back on, thereby connecting back into the job, the job can run again.
- The job is waiting for any reason. For example, when a job receives a reply to an inquiry message, the job can start running again.

JOBQ   The job is currently on a job queue. The job may have been previously active and was placed back on the job queue because of the Transfer Job (TFRJOB) or Transfer Batch Job (TFRBCHJOB) command, or the job was never active because it was just submitted.

OUTQ   The job has completed running and has spooled output that has not yet printed.

≫ *blank*   The job has no status. A possible reason is the job was not found.≪

Total length of data returned. The length of the data portion of the returned data. This includes the sum of all the keyed data for the job.

User name used. The user profile under which the job is run. The user name is the same as the user profile name and can come from several different sources depending on the type of job.

Format of Receiver Variable Definition Information

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of fields returned
These fields repeat, in the order listed, for each key selected.		BINARY(4)	Length of field information returned
		BINARY(4)	Key field
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data
		BINARY(4)	Displacement to data

Field Descriptions

Displacement to data. The displacement from the beginning of the job record to the value for this key.

Key field. The field returned. See Valid Keys in the List Job (QUSLJOB) API for the list of valid keys supported for the OLJB0200 format. See List of keys supported for format OLJB0300 for the list of valid

keys supported for the OLJB0300 format.

Length of data. The length of the data returned for the field.

Length of field information returned. The total length of definition information returned for this field. This value is used to increment to the next field in the list.

Number of fields returned. The number of fields in each record that is returned to the application.

Reserved. An ignored field.

Type of data. The type of data that is returned.

C   The data is returned in character format.

B   The data is returned in binary format.

Format of Sort Information

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of keys to sort on
Offsets vary. These fields repeat for each sort key field.		BINARY(4)	Sort key field starting position
		BINARY(4)	Sort key field length
		BINARY(2)	Sort key field data type
		CHAR(1)	Sort order
		CHAR(1)	Reserved

Field Descriptions

Number of keys to sort on. The number of fields within the record structure to sort on. If 0 is specified, the list is not sorted.

The following special value is supported for format OLJB0200 only. Also, the job selection information must be set to return only the jobs that are waiting to run from a single job queue.

-1   The list of jobs that are waiting to run on the specified job queue are returned in the order they will be selected off the job queue to become active. When this special value is used, the job queue priority (1005), data and time job was put on this job queue (0404), and the date and time job is scheduled to run (0403) keys must be included in the list of key fields to return.

The following special value is supported for format OLJB0300 only.

-2 The list of active jobs is returned grouped by subsystem, with each subsystem monitor job followed by the jobs that are running in that subsystem. The jobs are sorted alphabetically by subsystem name, and then by job name within the subsystem. This sort order is similar to the way the jobs are displayed on the Work with Active Jobs (WRKACTJOB SEQ(*SBS)) display. When this special value is used, the subsystem description name (1906) key must be included in the list of key fields to return.

Reserved. Reserved field. This field must be set to hexadecimal or binary zero.

Sort key field data type. Data type of field to sort. Refer to the [Sort](#) (QLGSORT) API for information on the list of data types available.

Sort key field length. The length of the field to sort on.

Sort key field starting position. Within the record of information, the starting position of the field to sort on. A value of 1 represents the first position within the record.

Sort order. Whether the list should be sorted in ascending or descending order according to the key.

1 Sort in ascending order.

2 Sort in descending order.

Format of Job Selection Information

The organization of the job selection information parameter is shown below. A description of the fields in the parameter follows the table.

OLJS0100 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Job name
10	0A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(1)	Job type
27	1B	CHAR(1)	Reserved (ignored)
28	1C	BINARY(4)	Offset to primary job status array
32	20	BINARY(4)	Number of primary job status entries
36	24	BINARY(4)	Offset to active job status array
40	28	BINARY(4)	Number of active job status entries
44	2C	BINARY(4)	Offset to jobs on job queue status array
48	30	BINARY(4)	Number of jobs on job queue status entries
52	34	BINARY(4)	Offset to job queue names array
56	38	BINARY(4)	Number of job queue names entries

The offsets to these fields are specified in the previous offset variables.	ARRAY(*) of CHAR(10)	Primary job status
	ARRAY(*) of CHAR(4)	Active job status
	ARRAY(*) of CHAR(10)	Jobs on job queue status
	ARRAY(*) of CHAR(20)	Job queue names

OLJS0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(60)	Everything in fixed portion of OLJS0100 format
60	3C	BINARY(4)	Offset to current user profile array
64	40	BINARY(4)	Number of current user profile entries
68	44	BINARY(4)	Offset to server type array
72	48	BINARY(4)	Number of server type entries
76	4C	BINARY(4)	Offset to the active subsystem array
80	50	BINARY(4)	Number of active subsystem entries
84	54	BINARY(4)	Offset to the memory pool array
88	58	BINARY(4)	Number of memory pool entries
92	5C	BINARY(4)	Offset to the job type - enhanced array
96	60	BINARY(4)	Number of job type - enhanced entries
» 100	64	BINARY(4)	Offset to the qualified job name array
104	68	BINARY(4)	Number of qualified job name entries «
The offsets to these fields are specified in the previous offset variables.		ARRAY(*)	Arrays from OLJS0100 format
		ARRAY(*) of CHAR(10)	Current user profile
		ARRAY(*) of CHAR(30)	Server type
		ARRAY(*) of CHAR(10)	Active subsystem
		ARRAY(*) of BINARY(4)	Memory pool
		ARRAY(*) of BINARY(4)	Job type - enhanced
		» ARRAY(*) of CHAR(26)	Qualified job name «

Field Descriptions

Active job status. The active status of the initial threads of the jobs to be included in the list. The possible values are the same as those described for the Retrieve Job Information (QUSRJOBI) API in the [Usage Notes](#) for that API. (For compatibility with previous releases, the CLDW status can be specified in the active job status array. The CLDW status is not returned, however, for any jobs.)

Active subsystem. The active subsystem under which the job is currently running.

Current user profile. The user profile under which the initial thread of the job is currently running. This name may differ from the user portion of the job name. The current user profile is blank for jobs that are not active.

Job name. A specific job name, a generic name, or one of the following special values:

*	Only the job in which this program is running. The user name and job name fields must be blank.
CURRENT	All jobs with the current job's name.
ALL	All jobs. The user name and job type fields must be specified.
» *blank*	This field must be blank when using the Qualified job name array.«

Job number. A specific job number or one of the following special values:

ALL	Jobs with the specified job name and user name, regardless of the job number. The job name and user name fields must be specified.
» *blank*	This field must be blank when using the Qualified job name array.«

Job queue names. The job queue names. Jobs on these job queues are included in the list. The first 10 characters contain the job queue name, and the last 10 characters contain the library name.

Jobs on job queue status. The status of the jobs on the job queue to be included in the list. The possible values follow:

SCD	This job will run as scheduled.
HLD	This job is being held on the job queue.
RLS	This job is ready to be selected.

Job type. The type of job to be listed. The possible values follow:

*	This value lists all job types.
A	The job is an autostart job.
B	The job is a batch job.
I	The job is an interactive job.
M	The job is a subsystem monitor job.
R	The job is a spooled reader job.

S   The job is a system job.

W   The job is a spooled writer job.

X   The job is the SCPF system job.

Job type - enhanced. The type of job to be listed. This field combines the Job type and Job Subtype fields. The possible values are:

0110   Autostart job.

0200   All the batch job types.

0210   Batch job.

0220   Batch immediate job.

0230   Batch - System/36 multiple requester terminal (MRT) job.

0240   Batch - Alternate spool user

0310   Communications job - procedure start request job.

0900   All interactive type jobs.

0910   Interactive job.

0920   Interactive job - Part of group.

0930   Interactive job - Part of system request pair.

0940   Interactive job - Part of system request pair and part of a group.

1600   All prestart jobs.

1610   Prestart job.

1620   Prestart batch job.

1630   Prestart communications job.

1810   Reader job.

1910   Subsystem job.

1920   System job.

2310   Writer job.

Memory pool. The identifier of the system memory pool in which the job started running. The identifier is a number in the range of 1 - 64.

Number of active job status entries. The number of entries that are specified for the active job status array. If this value is 0, no filtering is done on active job status. The offset to active job status array and the active job status array fields are not used. If the value is not 0, the primary job status array must include an array entry of *ACTIVE or the number of primary job status array entries should be specified as 0 to indicate that no filtering is done on the primary job status.

Number of active subsystem entries. The number of entries that are specified for the active subsystem

array. If this value is 0, no filtering is done on active subsystems, and the offset to the active subsystem array and the active subsystem array fields are not used. This value must be 0 when format OLJB0100 or OLJB0200 is specified.

Number of current user profile entries. The number of entries that are specified for the current user profile array. If this value is 0, no filtering is done on the current user profile, and the offset to current user profile array and the current user profile array fields are not used.

Number of job queue names entries. The number of entries that is specified for the job queue names array. If this value is 0, no filtering is done on the job queue names. The offset to job queue names array and the job queue names array fields are not used. If this value is not 0, the primary job status array must include an array entry of *JOBQ or the number of primary job status array entries should be specified as 0 to indicate that no filtering is done on the primary job status. This value must be 0 when format OLJB0300 is specified.

Number of jobs on job queue status entries. The number of entries that are specified for the jobs on job queue status array. If this value is 0, no filtering is done on the jobs on job queue status array. The offset to jobs on job queue status array and the jobs on job queue status array fields are not used. If this value is not 0, the primary job status array must include an array entry of *JOBQ or the number of primary job status array entries should be specified as 0 to indicate that no filtering is done on the primary job status. This value must be 0 when format OLJB0300 is specified.

Number of job type - enhanced entries. The number of entries that are specified for the job type - enhanced array. If this value is 0, no filtering is done on the job type - enhanced array. The offset to job type - enhanced array and the job type - enhanced array fields are not used.

Number of memory pool entries. The number of entries that are specified for the memory pool name array. If this value is 0, no filtering is done on memory pool, and the offset to the memory pool name array and the memory pool name array fields are not used. This value must be 0 when format OLJB0100 or OLJB0200 is specified.

Number of primary job status entries. The number of entries that are specified for the primary job status array. If 0 is specified, no filtering is done on the primary job status, and the offset to primary job status array and the primary job status array fields are not used. This value must be 0 when format OLJB0300 is specified.

≫**Number of qualified job name entries.** The number of entries that are specified for the qualified job name array. If 0 is specified, no filtering is done on the qualified job name, and the offset to qualified job name array and the qualified job name array fields are not used. This value must be 0 when format OLJB0300 is specified. This value must be 0 when anything other than blanks are specified in the job name, user name, or job number fields. The same number of jobs will be returned in the list of returned jobs. For jobs that are not found, the Status, Job type, and Job subtype fields will be set to blanks and the Job information status field will be set to 'L'.≪

Number of server type entries. The number of entries that is specified for the server type array. If this value is 0, no filtering is done on the server type. Also, the offset to server type array and the server type array fields are not used.

Offset to active job status array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the active job status array.

Offset to active subsystem array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the active subsystem array.

Offset to current user profile array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the current user profile array.

Offset to job queue names array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the job queue names array.

Offset to jobs on job queue status array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the jobs on job queue status array.

Offset to job type - enhanced array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the job type - enhanced array.

Offset to memory pool array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the memory pool name array.

Offset to primary job status array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the primary job status array.

»Offset to qualified job name array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the qualified job name array.«

Offset to server type array. The offset in characters (bytes) from the beginning of the job selection information structure to the beginning of the server type array.

Primary job status. The primary status of the jobs to be included in the list. The possible special values follow:

ACTIVE   Active jobs. This includes group jobs, system request jobs, and disconnected jobs.

JOBQ   Jobs that are currently on job queues.

OUTQ   Jobs that have completed running but still have output on an output queue.

»Qualified job name. The qualfied job name. The first 10 characters are the job name. The second 10 characters are the user name of the job. The last 6 characters are the job number.«

Reserved (ignored). An ignored field.

Reserved (must be 0). Reserved field. This field must be set to zero.

Server type. The type of server represented by the job. Servers provided by IBM start with QIBM. A server type, a generic value, or one of the following special values can be specified:

ALL   All jobs with a server type.

BLANK   All jobs without a server type.

User name. A specific user profile name, a generic name, or one of the following special values:

CURRENT   Jobs that use the current job's user profile.

ALL   Jobs that use the specified job name, regardless of the user name. The job name and job number fields must be specified.

» *blank*   This field must be blank when using the Qualified job name array.«

General Return Data

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(8), UNSIGNED	Elapsed time
16	10	CHAR(*)	Reserved

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.

Elapsed time. The time, in milliseconds, that has elapsed between the measurement start time and the current system time. This value is 0 the first time this API is called by this job. The measurement start is set the first time this API is called and when the Reset status statistics is set to reset the elapsed time.

Reserved. An ignored field.

List of keys supported for format OLJB0200

The list of possible keys is the same as that described for the List Job (QUSLJOB) API.

List of keys supported for format OLJB0300

Key	Type	Description
103	CHAR(4)	Active job status for jobs ending
305	CHAR(10)	Current user profile
312	BINARY(8), UNSIGNED	Processing unit time used - total for the job
313	BINARY(8), UNSIGNED	Processing unit time used for database - total for the job
314	BINARY(4)	Processing unit used - percent used during the elapsed time (job)
315	BINARY(8), UNSIGNED	Processing unit used - time during the elapsed time (job)
316	BINARY(4)	Processing unit used for database - percent used during the elapsed time (job)

317	BINARY(8), UNSIGNED	Processing unit used for database - time during the elapsed time (job)
402	CHAR(13)	Date and time job entered system
414	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time (job)
415	BINARY(8), UNSIGNED	Disk I/O count - total for the job
416	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - asynchronous I/O (job)
417	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - synchronous I/O (job)
502	CHAR(1)	End status
601	CHAR(10)	Function name
602	CHAR(1)	Function type
904	BINARY(4)	Interactive response time - total during the elapsed time
905	BINARY(4)	Interactive transactions - count during the elapsed time
1012	CHAR(10)	Job user identity
1014	BINARY(4)	Job end reason
1015	CHAR(1)	Job log pending
1016	BINARY(4)	Job type - enhanced
1306	CHAR(10)	Memory pool name
1307	CHAR(1)	Message reply
1609	BINARY(8), UNSIGNED	Page fault count during the elapsed time (job)
1802	BINARY(4)	Run priority (job)
1906	CHAR(20)	Subsystem description name - qualified
1911	CHAR(30)	Server type
≫1982	CHAR(10)	Spooled file action≪
2008	BINARY(4)	Thread count

Field Descriptions

The field descriptions are in [Work Management API Attribute Descriptions (WMAttrDesc)](#).

Usage Notes

The conditions under which this API is threadsafe are the same as those described in the [Usage Notes](#) for the Retrieve Job Information (QUSRJOBI) API.

Error Messages

Message ID	Error Message Text
CPF1865 E	Value &1 for job type not valid.
CPF1866 E	Value &1 for number of fields to return not valid.
CPF1867 E	Value &1 in list not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C39 E	Value for reserved field not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CB1 E	Value &1 for job status is not valid.
CPF3CB2 E	Value specified for job parameter is not valid.
CPF3CF1 E	Error code parameter not valid.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0024 E	&1 is not valid for number of keys to sort on.
GUI0025 E	&1 is not valid for sort key field starting position.
GUI0026 E	&1 is not valid for sort key field length.
GUI0027 E	&1 is not valid for number of records to return.
GUI0049 E	Key 1903 for status of jobs on job queues is not specified.
GUI0052 E	&1 is not valid for active status.
GUI0119 E	Reserved field in sort information not valid.
GUI0120 E	Order field in sort information not valid.
GUI0121 E	Printer name cannot be specified when format LSPL0100 is requested.
GUI0122 E	Number of primary job status entries not valid.
GUI0123 E	Active status key must be specified.
GUI0124 E	Primary status array must include *ACTIVE.
GUI0125 E	Status of job on job queue key must be specified.
GUI0126 E	Primary status array must include *JOBQ.
GUI0127 E	Number of job queue names not valid.
GUI0128 E	Job queue name key must be specified.

GUI0129 E   Number of active job status entries not valid.

GUI0130 E   Number of jobs on job queue status entries not valid.

GUI0131 E   &2 is not valid for length of job selection criteria.

GUI0132 E   &2 is not valid for length of receiver variable definition information.

GUI0133 E   Format OLJB0200 must be specified.

GUI0134 E   &2 not valid for length of job selection information.

GUI0137 E   Current user key must be specified.

GUI0138 E   Server type key must be specified.

GUI0139 E   Number of current users not valid.

GUI0140 E   Number of server types not valid.

》 GUI0142 E   Format OLJB0300 valid for active jobs only.

GUI0143 E   Number of memory pool names not valid.

GUI0144 E   Number of active subsystem names not valid.

GUI0145 E   All optional parameters must be specified.

GUI0146 E   Filter information not correct for format OLJB0200.

GUI0147 E   Required keys not specified.

GUI0148 E   Number of job type enhanced values not valid.

GUI0149 E   &1 is not valid for number of keys to sort on.

GUI0150 E   Subsystem key must be specified.

GUI0151 E   Job type enhanced key must be specified. 《

API introduced: V4R1

»Open List of Threads (QWCOLTHD) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Receiver variable definition information	Output	Char(*)
5	Length of receiver variable definition information	Input	Binary(4)
6	Job identification information	Input	Char(*)
7	Format of job identification information	Input	Char(8)
8	List Information	Output	Char(80)
9	Number of records to return	Input	Binary(4)
10	Sort information	Input	Char(*)
11	Number of fields to return	Input	Binary(4)
12	Key of fields to be returned	Input	Array(*) of Binary(4)
13	Reset status statistics	Input	Char(1)
14	General return data	Output	Char(*)
15	Length of general return data	Input	Binary(4)
16	Error Code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes.

The Open List of Threads (QWCOLTHD) API generates a list of active threads for the job specified in the Job identification parameter. The list may be sorted depending on the value of the sort parameter. Upon successful completion of this API, a handle is returned in the list information parameter. You may use this handle on subsequent calls to the following APIs:

> Get List Entries (QGYGTLE)
>
> Find List Entry (QGYFNDE)
>
> Close List (QGYCLST)

Authorities and Locks

Job Authority

> This API must be called from within the job for which the information is being retrieved or the caller of the API must be running under a user profile that is the same as the job user identity of the job for which the information is being retrieved or the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

> The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the [Work Management](#) book on the V5R1 Supplemental Manuals

Web site.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is used to return the list of active thread information.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable.

Format name

> INPUT; CHAR(8)

> The format of the thread list to be returned. The performance varies depending on the number of fields selected. The possible format names follow:

>> *OLTH0100*   List of active threads with keyed return fields.

> See [Format of Receiver Variable](#) for more information.

Receiver variable definition information

> OUTPUT; CHAR(*)

> The variable that is used to return the description of how the keyed portion of the list is returned in the receiver variable. See [Format of Receiver Variable Definition Information](#) for a description of the layout of this parameter.

Length of receiver variable definition information

> INPUT; BINARY(4)

> The length of the receiver variable definition information. This must be large enough to hold the information for all key fields that are specified in the key of fields to be returned parameter.

Job identification information

> INPUT; CHAR(*)

> The information that is used to identify the job for which the list of threads is to be returned. See [Format of job identification information](#) for details.

Format of job identification information

> INPUT; CHAR(8)

> The format of the job identification information. The possible format name is:

>> *JIDF0100*   See [Format of job identification information](#) for details on the job identification information.

List Information

> OUTPUT; CHAR(80)

Information about the list that is created by this program. For a description of the layout of this parameter, see [Format of open list information](#).

Number of records to return

    INPUT; BINARY(4)

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

Sort information

    INPUT; CHAR(*)

    Information on what fields within the record of information to sort. See [Format of Sort Information](#) for a description of the layout of this parameter.

Number of fields to return

    INPUT; BINARY(4)

    The number of fields to return. If this parameter is 0, the key of fields to be returned parameter is not used, the receiver variable definition information parameter is not modified, and the length of receiver variable definition information must be 0.

Key of fields to be returned

    INPUT; ARRAY(*) of BINARY(4)

    The list of the fields to be returned. For a list of the valid key fields, see [Valid Keys](#).

Reset status statistics

    INPUT; CHAR(1)

    The elapsed time and all the key fields that are based on the elapsed time are reset to zero. The following special values may be specified:

      0   The elapsed time and the key fields based on the elapsed time are not reset.

      1   The elapsed time and the key fields based on the elapsed time are reset back to zero.

General return data

    OUTPUT; CHAR(*)

    General output information that applies to the list of threads returned. See [General return data](#) for a description of the data returned.

Length of general return data

    INPUT; BINARY(4)

    The length of the general output information. The general output information field must be a minimum of 8 bytes.

Error code

    I/O; CHAR(*)

    The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Format of Receiver Variable

The following tables describe the order and format of the data that is returned in the receiver variable.

OLTH0100 Format

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(8)	Thread identifier
8	8	BINARY(4), UNSIGNED	Thread handle
12	C	BINARY(4)	Total length of data returned
16	F	CHAR(*)	Reserved
This information repeats for each key selected.		CHAR(*) or BINARY(4)	Data

Field Descriptions

Data. The data returned for the key field.

Reserved. An ignored field.

Thread handle. A value that is used to address a particular thread within a job. A valid thread handle must be specified. The thread handle is returned on several other interfaces.

Thread identifier. A value that is used to uniquely identify a thread within a job. A valid thread identifier must be specified.

Total length of data returned. The length of the data portion of the returned data. This includes the sum of all the keyed data for the list of threads.

Format of Receiver Variable Definition Information

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number of fields returned
These fields repeat, in the order listed, for each key selected.		BINARY(4)	Length of field information returned
		BINARY(4)	Key field
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data
		BINARY(4)	Displacement to data

Field Descriptions

Displacement to data. The displacement from the beginning of the thread record in the receiver variable to the value for this key.

Key field. The field returned. See [List of keys supported for format OLTH0100](#) for the list of valid keys supported for the OLTH0100 format.

Length of data. The length of the data returned for the field.

Length of field information returned. The total length of definition information returned for this field. This value is used to increment to the next field in the list.

Number of fields returned. The number of fields in each record that is returned to the application.

Reserved. An ignored field.

Type of data. The type of data that is returned.

- *C*   The data is returned in character format.
- *B*   The data is returned in binary format.

Format of Sort Information

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number of keys to sort on
Offsets vary. These fields repeat for each sort key field.		BINARY(4)	Sort key field starting position
		BINARY(4)	Sort key field length
		BINARY(2)	Sort key field data type
		CHAR(1)	Sort order
		CHAR(1)	Reserved

Field Descriptions

Number of keys to sort on. The number of fields within the record structure on which to sort. If 0 is specified, the list is not sorted.

Reserved. Reserved field. This field must be set to hexadecimal or binary zero.

Sort key field data type. Data type of field to sort. Refer to the [Sort](#) (QLGSORT) API for information on the list of data types available.

Sort key field length. The length of the field on which to sort.

Sort key field starting position. Within the record of information, the starting position of the field to sort on. A value of 1 represents the first position within the record.

Sort order. Whether the list should be sorted in ascending or descending order according to the key.

1   Sort in ascending order.

2   Sort in descending order.

Format of job identification information

The format of the information needed to identify the job for which the list of threads will be returned.

JIDF0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

*   The job that this program is running in. The job number and user name must contain blanks.

*INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hex zeros.

Thread identifier. An unused field on this API. This field must contain hex zeros.

Thread indicator. An unused field on this API. This field must contain hex zeros.

User name. A specific user profile name, or blanks when the job name specified is a special value.

General Return Data

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(8), UNSIGNED	Elapsed time
16	10	CHAR(10)	Job name used
26	1A	CHAR(10)	User name used
36	24	CHAR(6)	Job number used
42	2A	CHAR(16)	Internal job identifier
58	3A	CHAR(*)	Reserved

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.

Elapsed time. The time, in milliseconds, that has elapsed between the measurement start time and the current system time. This value is 0 the first time this API is called by this job. The measurement start is set the first time this API is called and when the Reset status statistics is set to reset the elapsed time.

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

Job name used. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the workstation where the job started. For a batch job, you specify the name in the command when you submit the job.

Job number used. The system-assigned job number.

Reserved. An ignored field.

User name used. The user profile under which the job is started. The user name is the same as the user profile name and can come from several different sources depending on the type of job.

List of keys supported for format OLTH0100

Key	Type	Description
305	CHAR(10)	Current user profile
319	BINARY(8), UNSIGNED	Processing unit time used - total for the thread
320	BINARY(8), UNSIGNED	Processing unit time used for database - total for the thread
321	BINARY(4)	Processing unit used - percent during the elapsed time (thread)
322	BINARY(8), UNSIGNED	Processing unit used - time during the elapsed time (thread)
323	BINARY(4)	Processing unit used for database - percent used during the elapsed time (thread)
324	BINARY(8), UNSIGNED	Processing unit time used for data base - time during the elapsed time (thread)
419	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time (thread)
420	BINARY(8), UNSIGNED	Disk I/O count - total for the thread
421	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - asynchronous I/O (thread)
422	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - synchronous I/O (thread)
1610	BINARY(8), UNSIGNED	Page fault count during the elapsed time (thread)
1804	BINARY(4)	Run priority (thread)
2010	CHAR(4)	Thread status
2011	CHAR(1)	Thread type

When the length of the data returned for a key is not a multiple of 4 bytes, the length of the field information returned is adjusted so that the next key begins on a 4-byte boundary. This adjustment should be considered when calculating the starting position of a key field to sort on.

Field Descriptions

The field descriptions are in Work Management API Attribute Descriptions (WMAttrDesc).

Error Messages

Message ID	Error Message Text
CPF136A E	Job &3/&2/&1 not active.
CPF1866 E	Value &1 for number of fields to return not valid.

CPF1867 E	Value &1 in list not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C3C E	Value for parameter &1 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.
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.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9F81 E	API &1 requires too much information to be collected prior to sorting.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0024 E	&1 is not valid for number of keys to sort on.
GUI0025 E	&1 is not valid for sort key field starting position.
GUI0026 E	&1 is not valid for sort key field length.
GUI0027 E	&1 is not valid for number of records to return.
GUI0119 E	Reserved field in sort information not valid.
GUI0120 E	Order field in sort information not valid.
GUI0132 E	&2 is not valid for length of receiver variable definition information.
GUI0149 E	&1 is not valid for number of keys to sort on.

《

API introduced: V5R2

Retrieve Call Stack (QWVRCSTK) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format of receiver information	Input	Char(8)
4	Job identification information	Input	Char(*)
5	Format of job identification information	Input	Char(8)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Call Stack (QWVRCSTK) API returns the call stack information for the specified thread. The first call stack entry returned corresponds to the most recent call in the thread.

Authorities and Locks

Job Authority

> The API must be called within the thread for which the call stack is being retrieved, or the caller of the API must be running under a user profile that is the same as the job user identity of the job containing the thread for which the call stack is being retrieved. Otherwise, the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

> The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the [Work Management](#) book on the V5R1 Supplemental Manuals Web site.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold. For example, this may mean that the value in the number of call stack entries returned field doesn't match the value in the number of call stack entries for thread field.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be

specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of receiver information

INPUT; CHAR(8)

The format of the information returned in the receiver variable. The possible format name is:

CSTK0100   See Format CSTK0100 for details on the call stack information returned.

Job identification information

INPUT; CHAR(*)

The information that is used to identify the thread within a job for which call stack information is to be returned. See Format of job identification information for details.

Format of job identification information

INPUT; CHAR(8)

The format of the job identification information. The possible format names are:

JIDF0100   See Format of job identification information for details on the job identification information.

≫ *JIDF0200*   See Format of job identification information for details on the job identification information.

Note: If the thread handle is available, Format JIDF0200 provides a faster method of accessing a thread that is not the current thread than Format JIDF0100. ≪

Error code

I/O; CHAR(*)

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

Format CSTK0100

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of call stack entries for thread
12	C	BINARY(4)	Offset to call stack entry information
16	10	BINARY(4)	Number of call stack entries returned
20	14	CHAR(8)	Returned thread identifier
28	1C	CHAR(1)	Information status
29	1D	CHAR(*)	Reserved

These fields repeat, in the order listed, for the number of call stack entries.	BINARY(4)	Length of this call stack entry
	BINARY(4)	Displacement to statement identifiers
	BINARY(4)	Number of statement identifiers
	BINARY(4)	Displacement to the procedure name
	BINARY(4)	Length of procedure name
	BINARY(4)	Request level
	CHAR(10)	Program name
	CHAR(10)	Program library name
	BINARY(4)	MI instruction number
	CHAR(10)	Module name
	CHAR(10)	Module library name
	CHAR(1)	Control boundary
	CHAR(3)	Reserved
	BINARY(4), UNSIGNED	Activation group number
	CHAR(10)	Activation group name
	» CHAR(2)	Reserved
	CHAR(10)	Program ASP name
	CHAR(10)	Program library ASP name
	BINARY(4)	Program ASP number
	BINARY(4)	Program library ASP number«
	CHAR(*)	Reserved
	ARRAY(*) of CHAR(10)	Statement identifiers
	CHAR(*)	Procedure name

Field Descriptions

Activation group name. The name of the activation group within which the program or procedure is running. Possible special values are:

DFTACTGRP   The activation group does not have a specific name. The activation group is one of the default activation groups for the system.

NEW   The activation group does not have a specific name. The activation group was created when the program was called.

Activation group number. The number of the activation group within which the program or procedure is running. This is an internal number that uniquely identifies the activation group within the job.

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.

Control boundary. Whether a control boundary is active for a particular program or procedure. Possible values are:

0 No control boundary is active.

1 A control boundary is active.

Displacement to procedure name. The displacement in bytes from the beginning of the call stack entry to the procedure name. This field is zero if the program at this call stack entry is not an ILE program.

Displacement to statement identifiers. The displacement in bytes from the beginning of the call stack entry to the array of statement identifiers. This field is zero if the number of statement identifiers is zero.

Information status. Whether the call stack entry information could be successfully retrieved.

blank No errors occurred. All information is returned in each entry.

I The information in each entry is not complete. The request level, control boundary, activation group number and activation group name fields could not be retrieved. The request level and activation group number are zero and the control boundary and activation group name are blank in each entry.

N The call stack entry information could not be retrieved. No entries are returned.

Length of procedure name. The length of the procedure name. This field is zero if the program at this call stack entry is not an ILE program.

Length of this call stack entry. The length of this call stack entry.

MI instruction number. The current machine instruction number in the program. This field is not meaningful for integrated language environment (ILE) procedures. A zero is returned for ILE procedures.

Module library name. The name of the library in which the module is located. The following special values may be returned:

*N The module library name is unavailable. Either the program has been destroyed or the library containing the program is locked.

blanks The program at this call stack entry is not an ILE program.

Module name. The module containing the integrated language environment (ILE) procedure. The following special values may be returned:

*N The module name is unavailable. Either the program has been destroyed or the library containing the program is locked.

blanks The program at this call stack entry is not an ILE program.

Number of call stack entries for thread. The number of call stack entries that are associated with this thread. A zero is returned if the call stack could not be retrieved.

Number of call stack entries returned. The number of call stack entries returned in the receiver variable.

Number of statement identifiers. The number of statement identifiers returned. This field is zero if the program or procedure was not created with debugging tables.

Offset to call stack entry information. The offset in bytes from the beginning of the receiver variable to

the first call stack entry.

Procedure name. The name of the procedure at this level of the call stack.

≫Program ASP name. The name of the auxiliary storage pool (ASP) device in which the program is located. The following special values also may be returned:

SYSBAS   The program is located in the system ASP or a basic user ASP.

N   The name of the ASP cannot be determined.

Program ASP number. The numeric identifier of the ASP containing the program. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP device cannot be determined.

Program library ASP name. The name of the ASP in which the program library is located. The following special values also may be returned:

SYSBAS   The program library is located in the system ASP or a basic user ASP.

N   The name of the ASP cannot be determined.

Program library ASP number. The numeric identifier of the ASP containing the program library. The following values may be returned:

1   The library is located in the system ASP or in a basic user ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP device cannot be determined.≪

Program library name. The name of the library in which the program is located. The following special values may be returned:

N   The program library name is unavailable. The library containing the program has been destroyed or is locked.

≫ *blanks*   The program is not located in a library.≪

Program name. The name of the program at this level of the call stack. ≫This can be any type of program object, including objects of type *PGM and *SRVPGM.≪ The following special value may be returned:

N   The program is unavailable. Either the program has been destroyed or the library containing the program is locked.

Request level. The level of the request-processing program or procedure. A zero is returned if the program or procedure has not received a request message.

Reserved. An unused field.

Returned thread identifier. A value which uniquely identifies the thread within the job.

Statement identifiers. The high-level language statement identifier. If this field contains the character representation of a number, the number is right-adjusted in the field and padded on the left with zeros (for example, '0000000246'). If the call stack entry is for an integrated language environment (ILE) procedure, more than one statement identifier may exist because of the compilers used for ILE languages.

Format of job identification information

The format of the information needed to identify the thread for which call stack information will be returned.

JIDF0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

    *   The job in which this program is running. The job number and user name must contain blanks.

  *INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread identifier. A value which uniquely identifies a thread within a job. If the thread indicator is not 0, this field must contain hex zeros.

Thread indicator. A value which is used to specify the thread within the job for which information is to be retrieved. The following values are supported:

0   Specifies that information should be retrieved for the thread specified in the thread identifier field.

1   Specifies that information should be retrieved for the thread that this program is currently running in. The combination of the internal job identifier, job name, job number, and user name fields must also identify the job containing the current thread.

2   Specifies that information should be retrieved for the initial thread of the identified job.

User name. A specific user profile name, or blanks when the job name specified is a special value.

»JIDF0200 Format

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4), UNSIGNED	Thread handle
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

*   The job in which this program is running. The job number and user name must contain blanks.

INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread handle. A value which addresses a particular thread within a job. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread. A valid thread handle must be specified. The thread handle is returned on several other interfaces.

Thread identifier. A value which uniquely identifies a thread within a job. A valid thread identifier must

be specified.

User name. A specific user profile name, or blanks when the job name specified is a special value.«

Error Messages

Message ID	Error Message Text
CPF136A E	Job &3/&2/&1 not active.
CPF18BF E	Thread &1 not found.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
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.
CPF3C3B E	Value for parameter &2 for API &1 not valid
CPF3C3C E	Value for parameter &1 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.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

Retrieve Class Information (QWCRCLSI) API

```
Required Parameter Group:


    1    Receiver variable              Output      Char(*)
    2    Length of receiver variable    Input       Binary(4)
    3    Format of class information     Input       Char(8)
    4    Qualified class name            Input       Char(20)
    5    Error Code                      I/O         Char(*)


Default Public Authority: *USE

Threadsafe: No
```

The Retrieve Class Information (QWCRCLSI) API returns the attributes of a class object. A class contains the job run attributes for jobs that use this class. This API provides support similar to the Display Class (DSPCLS) command.

Authorities and Locks

Class

     *USE

Class Library

     *EXECUTE

Required Parameter Group

Receiver variable

    OUTPUT; CHAR(*)

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

Length of receiver variable

    INPUT; BINARY(4)

    The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of class information

    INPUT; CHAR(8)

The format of the class information being returned. The format names that can be used are as follows:

CLSI0100   This format returns all of the class information. See Format CLSI0100 for details.

Qualified class name

INPUT; CHAR(20)

The class name whose attributes are to be retrieved. The first 10 characters contain the class name, and the second 10 characters contain the library name. You can use these special values for the library name:

*CURLIB   The job's current library

*LIBL   The job's library list

Error code

I/O; CHAR(*)

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

Format CLSI0100

The following information is returned by this API when format CLSI0100 is used:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Class name
18	12	CHAR(10)	Class library name
28	1C	BINARY(4)	Run priority
32	20	BINARY(4)	Time slice
36	24	BINARY(4)	Eligible for purge
40	28	BINARY(4)	Default wait time
44	2C	BINARY(4)	Maximum CPU time
48	30	BINARY(4)	Maximum temporary storage in kilobytes
52	34	BINARY(4)	Maximum number of threads
56	38	CHAR(50)	Text
106	6A	CHAR(2)	Reserved
108	6C	BINARY(4)	Maximum temporary storage in megabytes

Field Description

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.

Class library name. The name of the library in which the class resides.

Class name. The name of the class about which information is returned.

Default wait time. The default maximum time (in seconds) that a thread in a job waits for a system instruction, such as the LOCK machine interface (MI) instruction, to acquire a resource. This default wait time is used when a wait time is not otherwise specified for a given situation. Normally, this would be the amount of time the user would be willing to wait for the system before the request is ended. A value of -1 is returned for *NOMAX, which indicates there is no maximum wait time.

Eligible for purge. Whether or not the job is eligible to be moved out of main storage and put into auxiliary storage at the end of a time slice or when beginning a long wait (such as waiting for a work station user's response). This attribute is ignored when more than one thread is active within the job. The possible values are:

0 The job is not eligible to be moved out of main storage and put into auxiliary storage. When main storage is needed, however, pages belonging to a thread in this job may be moved to auxiliary storage. Then, when a thread in the job runs again, its pages are returned to main storage as they are needed.

1 The job is eligible to be moved out of main storage and put into auxiliary storage. A job with multiple threads, however, is never purged from main storage.

Maximum CPU time. The maximum processing unit time (in milliseconds) that the job can use. If the job consists of multiple routing steps, this is the maximum processing unit time that the routing step can use. If the maximum time is exceeded, the job is ended. A value of -1 is returned for *NOMAX, which indicates there is no limit on the processing unit time.

Maximum number of threads. The maximum number of threads that a job using this class can run with at any time. If multiple threads are initiated simultaneously, this value may be exceeded. If this maximum value is exceeded, the excess threads will be allowed to run to their normal completion. Initiation of additional threads will be inhibited until the maximum number of threads in the job drops below this maximum value. A value of -1 is returned for *NOMAX, which indicates there is no maximum number of threads. Depending on the resources used by the threads and the resources available on the system, the initiation of additional threads may be inhibited before the maximum is reached.

Maximum temporary storage in kilobytes. The maximum amount of auxiliary storage (in kilobytes) that the job can use. If the job consists of multiple routing steps, this is the maximum temporary storage that the routing step can use. This temporary storage is used for storage that is required by the programs running in the job and by internal system objects created while the programs are running. (It does not include storage in the QTEMP library.) If the maximum temporary storage is exceeded, the job is ended. This does not apply to the use of permanent storage, which is controlled through the user profile. A value of -1 is returned for *NOMAX, which indicates the system maximum is used.

Maximum temporary storage in megabytes. The maximum amount of auxiliary storage (in megabytes) that the job can use. If the job consists of multiple routing steps, this is the maximum temporary storage that the routing step can use. This temporary storage is used for storage that is required by the programs running in the job and by internal system objects created while the programs are running. (It does not include storage in the QTEMP library.) If the maximum temporary storage is exceeded, the job is ended. This does not apply to the use of permanent storage, which is controlled through the user profile. A value of -1 is

returned for *NOMAX, which indicates the system maximum is used.

Reserved. This field is ignored.

Run priority. A value that represents the priority at which the job competes for the processing unit relative to other jobs that are active at the same time. The run priority ranges from 0 (highest priority) to 99 (lowest priority). This value is the highest run priority allowed for any thread within the job. Individual threads may have a lower priority.

Text. The text description of the class.

Time slice. The maximum amount of processor time, in milliseconds, given to each thread in a job before other threads in the job or other jobs are given the opportunity to run. The time slice establishes the amount of time needed by a thread in the job to accomplish a meaningful amount of processing. At the end of the time slice, the thread might be put in an inactive state so that other threads can become active in the storage pool.

Error Messages

Message ID	Error Message Text
CPF1029 E	No authority to library &1.
CPF1039 E	Class library &1 not found.
CPF1065 E	Class &1 in library &2 not found.
CPF1067 E	Cannot allocate library &1.
CPF1068 E	Cannot allocate class &1 in library &2.
CPF1098 E	No authority to class &1 in library &2.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R3

Retrieve Current Attributes (QWCRTVCA) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Number of attributes to return	Input	Binary(4)
5	Key of attributes to be returned	Input	Array(*) of Binary(4)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Conditional; see [Valid Key Attributes](#).

The Retrieve Current Attributes (QWCRTVCA) API retrieves job and thread attributes that apply to the thread in which this API is called. If a thread attribute exists, it is retrieved. If a thread attribute does not exist, the job attribute for the job in which this thread is running is retrieved.

Authorities and Locks

None.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is used to return the thread attributes.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

Format name

INPUT; CHAR(8)

The format of the attribute list to return. The possible format names follow:

≫ *RTVC0100*   Basic retrieve format

RTVC0200   Library list information

RTVC0300   ASP group information

See

Number of attributes to return

INPUT; BINARY(4)

The number of attributes to return in the specified format.

Key of attributes to be returned

INPUT; ARRAY(*) of BINARY(4)

The list of the attributes to be returned in the specified format. For a list of the valid key attributes, see Valid Key Attributes.

Error code

I/O; CHAR(*)

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

RTVC0100 Format

The following table describes the order and format of the data that is returned in the receiver variable for format RTVC0100. See Valid Key Attributes for threadsafe information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of attributes returned
These fields repeat, in the order listed, for each key requested.		BINARY(4)	Length of attribute information returned
		BINARY(4)	Key
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data
		CHAR(*)	Data
		CHAR(*)	Reserved

Field Descriptions

Data. The data returned for the key field.

Key. The attribute returned. See Valid Key Attributes for the list of valid keys.

Length of attribute information returned. The total length of information returned for this attribute. This value is used to increment to the next entry in the list.

Length of data. The length of the data returned for the field.

Number of attributes returned. The number of attributes returned to the application.

Reserved. An ignored field.

Type of data. The type of output data. This field is provided to maintain the same format layout that is used in the Change Job (QWTCHGJB) API.

C   The output data is in character format.

B   The output data is in binary format.

RTVC0200 Format

The RTVC0200 format returns library list information. Retrieval of the library list information is threadsafe. The format returns the actual length instead of the total length because all libraries may not exist.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	BINARY(4)	Number of libraries in SYSLIBL
12	C	BINARY(4)	Number of product libraries
16	10	BINARY(4)	Current library existence
20	14	BINARY(4)	Number of libraries in USRLIBL
See note	See note	Array(*) of CHAR(11)	Library list (for each library in the list)

Note: The decimal and hexadecimal offsets depend on the number of libraries you have in the various parts of your library lists and on keys requested. The data is left-justified and padded with blanks on the right. The array is sequential. See the

CL Programming book for the total number of libraries that can be returned to you.

Field Descriptions

Current library existence. Whether the current library exists or not. This value will be zero if the current library was not requested.

0   No current library exists.

1   A current library exists.

Number of bytes available. All of the available bytes for use in your application.

The actual length depends on how many libraries are in the library list.

Number of bytes returned. The number of bytes returned to the user. This may be some but not all of the bytes available.

Number of libraries in SYSLIBL. The number of libraries in the system part of the thread's library list. This value will be zero if system libraries were not requested.

Number of libraries in USRLIBL. The number of libraries in the thread's user library list. This value will be zero if user libraries were not requested.

Number of product libraries. The number of product libraries found in the thread's library list. This value will be zero if product libraries were not requested.

Library list (for each library in the list). The list of all libraries requested. A blank is in the last position of each name. The number of libraries in the list will depend on the keys requested and the actual number of libraries in each portion of the library list. The order of the library list is:

- System library list
- Product libraries
- Current library
- User library list

⟫RTVC0300 Format

The RTVC0300 format returns auxiliary storage pool (ASP) group information. Retrieval of the ASP group information is threadsafe.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	BINARY(4)	Offset to ASP group information
12	C	BINARY(4)	Number of entries in ASP group information
16	10	BINARY(4)	Length of one ASP group information entry
See note	See note	Array(*) of CHAR(*)	ASP group information entry (See Format of ASP Group Information Entry for more information.)
Note: This field repeats for each ASP group information entry. For additional information, see the Control Language (CL) information for the Set ASP Group (SETASPGRP) command.			

Field Descriptions

ASP group information entry. The auxiliary storage pool (ASP) group information requested. This does not include the system ASP or the basic user ASP.

Length of one ASP group information entry. The length of one ASP group information entry.

Number of ASP group information entries The number of ASP group information entries being returned.

Number of bytes available. The number of available bytes for use by your application.

Number of bytes returned. The number of bytes returned to the user. This may be some but not all of the bytes available.

Offset to ASP group information The offset in characters (bytes) from the beginning of the receiver to the first ASP group information entry.

Format of ASP Group Information Entry

The ASP group information entry describes the data that is returned for each ASP group of the RTVC0300 format.

Offset		Type	Field
Dec	Hex		
These fields repeat for each entry returned in the list.		CHAR(10)	ASP group name
		CHAR(*)	Reserved

Field Descriptions

ASP group name The name of the ASP group. This is the name of the primary ASP device in an ASP group.

≪

Valid Key Attributes

The following table contains a list of the valid keys for format RTVC0100. In addition, the table indicates whether the attributes are threadsafe. See Key Field Descriptions for the descriptions of the valid key attributes.

Key	Type	Description	Threadsafe (See Note)
0102	CHAR(1)	Allow multiple threads	Yes
0201	CHAR(10)	Break message handling	Yes
0301	CHAR(1)	Cancel key	Yes
0302	BINARY(4)	Coded character set ID	Yes
0303	CHAR(2)	Country or region ID	Yes
0305	CHAR(10)	Current user profile	Yes
0307	BINARY(4)	Current system pool identifier	Yes
0311	CHAR(10)	Character identifier control	Yes
0401	CHAR(13)	Date and time job became active	Yes
0402	CHAR(13)	Date and time job entered system	Yes
0403	CHAR(8)	Date and time job was scheduled to run	Yes
0405	CHAR(4)	Date format	Yes
0406	CHAR(1)	Date separator	Yes

0407	CHAR(1)	DBCS-capable	Yes
0408	CHAR(10)	DDM conversation handling	Yes
0409	BINARY(4)	Default wait	Yes
0410	CHAR(13)	Device recovery action	Yes
0412	BINARY(4)	Default coded character set identifier	Yes
0413	CHAR(1)	Decimal format	Yes
0501	BINARY(4)	End severity	Yes
0502	CHAR(1)	End status	Yes
0503	CHAR(1)	Exit key	Yes
0702	CHAR(10)	Group profile name	Yes
0703	CHAR(150)	Group profile name - supplemental	Yes
0901	CHAR(10)	Inquiry message reply	Yes
0902	CHAR(16)	Internal job ID	Yes
0903	CHAR(1)	Initial thread	Yes
1001	CHAR(15)	Job accounting code	Yes
1002	CHAR(7)	Job date	Yes
1004	CHAR(20)	Job queue name - qualified	Yes
1005	CHAR(2)	Job queue priority	Yes
1006	CHAR(8)	Job switches	Yes
1007	CHAR(10)	Job message queue full action	Yes
1008	BINARY(4)	Job message queue maximum size	Yes
1009	CHAR(26)	Job name	Yes
1010	CHAR(1)	Job type	Yes
1011	CHAR(1)	Job subtype	Yes
1201	CHAR(3)	Language ID	Yes
1202	CHAR(1)	Logging level	Yes
1203	CHAR(10)	Logging of CL programs	Yes
1204	BINARY(4)	Logging severity	Yes
1205	CHAR(10)	Logging text	Yes
1304	BINARY(4)	Maximum threads	Yes
1501	CHAR(20)	Output queue name - qualified	Yes
1502	CHAR(2)	Output queue priority	Yes
1601	CHAR(10)	Print key format	Yes
1602	CHAR(30)	Print text	Yes
1603	CHAR(10)	Printer device name	Yes
1604	CHAR(10)	Purge	Yes
1802	BINARY(4)	Run priority	Yes
1901	CHAR(20)	Sort sequence table - qualified	Yes
1902	CHAR(10)	Status message handling	Yes
1904	CHAR(26)	Submitter's job name - qualified	Yes
1905	CHAR(20)	Submitter's message queue name - qualified	Yes
1907	BINARY(4)	System pool identifier	Yes

	1982	CHAR(10)	Spooled file action	Yes
2001	CHAR(1)	Time separator	Yes
2002	BINARY(4)	Time slice	Yes
2003	CHAR(10)	Time-slice end pool	Yes
2008	BINARY(4)	Thread count	Yes

Note: If this value is blank, the attribute is not threadsafe.

The following table contains a list of the valid keys for format RTVC0200. In addition, the table indicates whether the attributes are threadsafe. See [Key Field Descriptions](#) for the descriptions of the valid key attributes.

Key	Type	Description	Threadsafe (See Note)
0310	CHAR(11)	Current library	Yes
1660	Array(*) of CHAR(11)	Product libraries	Yes
1980	Array(*) of CHAR(11)	System library list	Yes
2110	Array(*) of CHAR(11)	User library list	Yes
2702	Array(*) of CHAR(11)	All portions of the library list for format RTVC0200	Yes

Note: If this value is blank, the attribute is not threadsafe.

The following table contains a list of the valid keys for format RTVC0300. In addition, the table indicates whether the attributes are threadsafe. See [Key Field Descriptions](#) for the descriptions of the valid key attributes.

Key	Type	Description	Threadsafe (See Note)
0104	CHAR(*)	ASP group information	Yes

Key Field Descriptions

Most field descriptions for this API are in [Work Management API Attribute Descriptions (WMAttrDesc)](#), except the following:

All portions of the library list for format RTVC0200. All portions of the library list will be returned.

Current system pool identifier. The identifier of the system-related pool from which this thread's main storage currently is being allocated. These identifiers are not the same as those specified in the subsystem description, but are the same as the system pool identifiers shown on the system status display. If a thread reaches the end of its time slice, the pool this thread is running in can be switched based on the job's time-slice end pool value. The current system pool identifier returned by this API will be the actual pool in which the thread currently is running.

Run priority. The priority at which this thread currently is running, relative to other threads on the system. The run priority ranges from 0 (highest priority) to 99 (lowest priority). The value may never be higher than the run priority for the job in which this thread is running. Since this API is intended for retrieving the current value of an attribute, the run priority of the thread is returned, even though the 1802 key represents the run priority of the job on other interfaces. To obtain the run priority of the job, use the

1802 key on the [Retrieve Thread Attributes](#) (QWTRTVTA) API.≪

Error Messages

Message ID	Error Message Text
CPF1866 E	Value &1 for number of fields to return not valid.
CPF1867 E	Value &1 in list not valid.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
≫ 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: V4R2

Retrieve Data Area (QWCRDTAA) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Qualified data area name	Input	Char(20)
4	Starting position	Input	Binary(4)
5	Length of data	Input	Binary(4)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes

The Retrieve Data Area (QWCRDTAA) API allows you to retrieve the contents of a data area. Distributed data management (DDM) data areas are supported by this API. In other words, this API can retrieve a data area value from a data area that exists on a remote OS/400.

Authorities and Locks

Library Authority
> *EXECUTE

Data Area Authority
> *USE

Data Area Lock
> *SHRRD

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

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

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable described in Format of Data Returned. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8

bytes.

Qualified data area name

INPUT; CHAR(20)

The first 10 characters contain the data area name, and the second 10 characters contain the name of the library where the data area is located.

When one of the special values is specified, the library name must be blank. The special values for the data area are:

LDA Local data area

GDA Group data area

PDA Program initialization parameter data area

The special values supported for the library name are:

LIBL   The library list.

CURLIB The job's current library.

Starting position

INPUT; BINARY(4)

The first byte of the data area to be retrieved. A value of 1 will identify the first character in the data area. The maximum value allowed for the starting position is 2000. A value of -1 will return all the characters in the data area.

Length of data

INPUT; BINARY(4)

The length of the data area substring to be retrieved. The length of data parameter must be greater than 0. If the length of data parameter is greater than the size of the data area, the receiver variable is padded with blanks.

The length of data parameter added to the starting position minus one must be between the substring starting position and the data area end; otherwise, CPF1089 (Substring specified for data area not valid) is issued. If you have a small data area and only want the fixed portion of the data returned, you must code -1 for the starting position.

Error code

I/O; CHAR(*)

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

Format of Data Returned

The receiver variable holds the information returned for the data area. The following table shows the format of the receiver variable.

Offset		

Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes available
4	4	BINARY(4)	Bytes returned
8	8	CHAR(10)	Type of value returned
18	12	CHAR(10)	Library name
28	1C	BINARY(4)	Length of value returned
32	20	BINARY(4)	Number of decimal positions
36	24	CHAR(*)	Value

Field Descriptions

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of all data actually returned. If the data is truncated because the receiver variable was not sufficiently large to hold all of the data available, this value will be less than the bytes available.

Length of value returned. The length of the value that was returned.

Library name. The name of the library where the data area was found. This field will be blank if one of the special values was specified for the first ten characters of the qualified data area name.

Number of decimal positions. The number of decimal positions.

Type of value returned. The following values may be returned.

CHAR   A character data area.

DEC   A decimal data area. The value returned will be a packed decimal value.

LGL   A logical data area.

Value. The contents of the data area.

Usage Notes

This API is threadsafe, except in the following situations:
- The retrieval of DDM data areas in a job that allows multiple threads is not threadsafe.
- The retrieval of DDM data areas will not be allowed when more than one thread is active in a job.

Error Messages

Message ID	Error Message Text
CPF101A E	Operation on DDM data area &1 in &2 failed.
CPF1015 E	Data area &1 in &2 not found.
CPF1016 E	No authority to data area &1 in &2.
CPF1021 E	Library &1 not found for data area &2.
CPF1022 E	No authority to library &1 data area &2.
CPF1046 E	DTAARA(*GDA) not valid because job not group job.
CPF1063 E	Cannot allocate data area &1 in library &2.
CPF1067 E	Cannot allocate library &1.
CPF1072 E	DTAARA(*PDA) not valid because job not prestart job.
CPF1088 E	Starting position outside of data area.
CPF1089 E	Substring specified for data area not valid.
CPF180B E	Function &1 not allowed.
CPF1863 E	Length of value not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
≫ CPF3CF2 E	Error(s) occurred during running of &1 API.≪
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.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Retrieve IPL Attributes (QWCRIPLA) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format of IPL attributes	Input	Char(8)
4	Error Code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve IPL Attributes (QWCRIPLA) API returns the settings of attributes that are used during the IPL. This API provides support similar to the Display IPL Attributes (DSPIPLA) command.

Authorities and Locks

None.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

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

Length of receiver variable

INPUT BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of IPL attributes

INPUT CHAR(8)
The format of the IPL attributes being returned. The format names that can be used are as follows:

IPLA0100   This format returns all of the IPL attributes. See format IPLA0100 for details.

Error code

I/O; CHAR(*)

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

Format IPLA0100

The following information is returned by this API when format IPLA0100 is used:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(1)	Restart type
9	9	CHAR(1)	Keylock position
10	A	CHAR(1)	Hardware diagnostics
11	B	CHAR(1)	Compress job tables
12	C	CHAR(1)	Check job tables
13	D	CHAR(1)	Rebuild product directory
14	E	CHAR(1)	Mail Server Framework recovery
15	F	CHAR(1)	Clear job queues
16	10	CHAR(1)	Clear output queues
17	11	CHAR(1)	Clear incomplete job logs
18	12	CHAR(1)	Start print writers
19	13	CHAR(1)	Start system to restricted state
20	14	CHAR(1)	Display status
21	15	CHAR(1)	Start TCP/IP

Field Description

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.

Check job tables. When to perform particular damage checks on the job tables.

0   The job table checks are performed during abnormal IPLs only.

1   The job table checks are performed during all IPLs.

2   The job table checks are performed synchronously during all IPLs.

Clear incomplete job logs. Whether or not to delete the job logs for jobs that were active at the time of the last system power down. This value is reset to 0 after each IPL.

0   The job logs are produced after the IPL.

1   The job logs are deleted during the IPL.

Clear job queues. Whether or not to clear the jobs from all job queues. This value is reset to 0 after each IPL.

0   The job queues are not cleared.

1   The job queues are cleared during the IPL.

Clear output queues. Whether or not to clear all output queues. If you clear the output queues, all spooled output is removed from the system. This value is reset to 0 after each IPL.

0   The output queues are not cleared.

1   The output queues are cleared during the IPL.

Compress job tables. When the job tables should be compressed to remove excess unused entries.

0   The job tables are compressed during abnormal IPLs only.

1   The job tables are compressed during all IPLs.

2   The job tables are not compressed during any IPL.

3   The job tables are compressed during normal IPLs only.

Display status. When the status of OS/400 IPL steps is displayed on the console during IPL. Status is not displayed during install IPLs or when the console is not powered on.

0   Status is displayed during attended OS/400 IPLs and abnormal OS/400 IPLs.

1   Status is not displayed during OS/400 IPLs.

2   Status is displayed during attended OS/400 IPLs.

3   Status is displayed during abnormal OS/400 IPLs.

4   Status is displayed during all OS/400 IPLs, except as noted above.

Hardware diagnostics. Whether or not certain hardware diagnostics should be performed during the IPL.

0   All hardware diagnostics are run.

1   The minimum set of hardware diagnostics is run.

Keylock position. The keylock position.

0   The keylock position is set to auto.

1   The keylock position is set to manual.

2   The keylock position is set to normal.

3   The keylock position is set to secure.

Mail Server Framework recovery. Whether or not Mail Server Framework recovery should be done

during IPL. The possible values are:

0   Mail Server Framework recovery is not done during IPL. Recovery is done when Mail Server Framework starts.

1   Mail Server Framework recovery is done during abnormal IPLs.

Rebuild product directory. When the product directory information is rebuilt.

0   The product directory information is rebuilt during abnormal IPLs only.

1   The product directory information is rebuilt during all IPLs.

2   The product directory information is not rebuilt during IPL.

3   The product directory information is rebuilt during normal IPLs only.

Restart type. The type of restart operation to perform when the Power Down System (PWRDWNSYS) command is used with RESTART(*YES).

0   All portions of the system, including the hardware, are restarted.

1   The operating system is restarted. The hardware is restarted only if you apply a PTF that requires a hardware restart. This value can reduce the time required to restart the system.

Start print writers. Whether or not print writers should be started at the time of the IPL. This value is reset to 1 after each IPL.

0   The print writers are not started at IPL time.

1   The print writers are started at IPL time.

Start system to restricted state. Whether or not the system should be started in the restricted state. If the system is started in the restricted state, only the system console is active. This value is reset to 0 after each IPL.

0   The system is not started in the restricted state.

1   The system is started in the restricted state.

Start TCP/IP. Whether the STRTCP command is submitted automatically at the completion of IPL and when the controlling subsystem is restarted from the restricted state. The STRTCP command is not submitted during install IPLs or when the system is starting to the restricted state. See the STRTCP command help for more information.

0   The system does not automatically submit the STRTCP command at the completion of IPL.

1   The system automatically submits the STRTCP command at the completion of IPL.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Retrieve Job Description Information (QWDRJOBD) 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 description name  Input      Char(20)
    5    Error code                     I/O        Char(*)


Default Public Authority: *USE

Threadsafe: No
```

The Retrieve Job Description Information (QWDRJOBD) API retrieves information from a job description object and places it into a single variable in the calling program. The amount of information returned depends on the size of the variable. The information returned is the same information returned by the Display Job Description (DSPJOBD) command.

Authorities and Locks

Job Description Object Authority

> *USE

Library Authority

> *EXECUTE

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 format of the job description information to be returned. You can use this format:

JOBD0100   Basic job description information. For details, see JOBD0100 Format.

Qualified job description name

INPUT; CHAR(20)

The name of the job description whose contents are to be retrieved. The first 10 characters contain the name of the job description, and the second 10 characters contain the name of the library where the job description is located. You can use these special values for the library name:

CURLIB   The job's current library

LIBL   The library list

Error code

I/O; CHAR(*)

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

JOBD0100 Format

The following table describes the information that is returned in the receiver variable for the JOBD0100 format. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Job description name
18	12	CHAR(10)	Job description library name
28	1C	CHAR(10)	User name
38	26	CHAR(8)	Job date
46	2E	CHAR(8)	Job switches
54	36	CHAR(10)	Job queue name
64	40	CHAR(10)	Job queue library name
74	4A	CHAR(2)	Job queue priority
76	4C	CHAR(10)	Hold on job queue
86	56	CHAR(10)	Output queue name
96	60	CHAR(10)	Output queue library name
106	6A	CHAR(2)	Output queue priority
108	6C	CHAR(10)	Printer device name
118	76	CHAR(30)	Print text
148	94	BINARY(4)	Syntax check severity

Dec	Hex	Type	Field
152	98	BINARY(4)	End severity
156	9C	BINARY(4)	Message logging severity
160	A0	CHAR(1)	Message logging level
161	A1	CHAR(10)	Message logging text
171	AB	CHAR(10)	Logging of CL programs
181	B5	CHAR(10)	Inquiry message reply
191	BF	CHAR(13)	Device recovery action
204	CC	CHAR(10)	Time-slice end pool
214	D6	CHAR(15)	Accounting code
229	E5	CHAR(80)	Routing data
309	135	CHAR(50)	Text description
359	167	CHAR(1)	Reserved
360	168	BINARY(4)	Offset to initial library list
364	16C	BINARY(4)	Number of libraries in initial library list
368	170	BINARY(4)	Offset to request data
372	174	BINARY(4)	Length of request data
376	178	BINARY(4)	Job message queue maximum size
380	17C	CHAR(10)	Job message queue full action
390	186	CHAR(10)	CYMD job date
400	190	CHAR(10)	Allow multiple threads
» 410	19A	CHAR(10)	Spooled file action
420	1A4	BINARY(4)	Offset to initial ASP group information
424	1A8	BINARY(4)	Number of initial ASP group information entries
428	1AC	BINARY(4)	Length of one initial ASP group information entry«
432	1B0	CHAR(*)	Reserved
*	*	ARRAY (*) of CHAR(11)	Initial library list
*	*	CHAR(*)	Request data
»		Array(*) of CHAR(*)	Initial ASP group information entry

Format of Initial ASP Group Information Entry

The initial auxiliary storage pool (ASP) group information entry describes the data that is returned for each group in the job description's initial ASP group.

Offset			
Dec	Hex	Type	Field
The fields repeat for each entry returned in the initial ASP group information.		CHAR(10)	ASP group name
		CHAR(*)	Reserved«

Field Descriptions

Accounting code. An identifier assigned to jobs that use this job description. This code is used to collect system resource use information. If the special value *USRPRF is specified, the accounting code used for jobs using this job description is obtained from the job's user profile.

Allow multiple threads. Whether or not the job is allowed to run with multiple user threads. This attribute does not prevent the operating system from creating system threads in the job. The possible values are *YES and *NO. This attribute is not allowed to be changed once a job starts. This attribute applies to autostart jobs, prestart jobs, batch jobs submitted from job schedule entries, and jobs started by using the Submit Job (SBMJOB) and Batch Job (BCHJOB) commands. This attribute is ignored when starting all other types of jobs. This attribute should be set to *YES only in job descriptions that are used exclusively with functions that create multiple user threads.

»ASP group name. The name of the ASP group. This is the name of the primary ASP device in an ASP group or the name of an ASP device description. This specifies the initial ASP group setting for jobs using this job description. «

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of all data actually returned. If the data is truncated because the receiver variable was not sufficiently large to hold all of the data available, this value will be less than the bytes available.

CYMD job date. The date that will be assigned to jobs using this job description when they are started. The possible values are:

*SYSVAL	The value in the QDATE system value is used at the time the job is started.
job-date	The date to be used at the time the job is started. The format of the field returned in CYYMMDD where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century indicates years 19xx and a 1 indicates years 20xx. The field is padded on the right with blanks.

Device recovery action. The action to take when an I/O error occurs for the interactive job's requesting program device. The possible values are:

*SYSVAL	The value in the system value QDEVRCYACN at the time the job is started is used as the device recovery action for this job description.
*MSG	Signals the I/O error message to the application and lets the application program perform error recovery.
*DSCMSG	Disconnects the job when an I/O error occurs. When the job reconnects, the system sends a message to the application program, indicating the job has reconnected and that the workstation device has recovered.
*DSCENDRQS	Disconnects the job when an I/O error occurs. When the job reconnects, the system sends the End Request (ENDRQS) command to return control to the previous request level.

*ENDJOB	Ends the job when an I/O error occurs. A message is sent to the job's log and to the history log (QHST). This message indicates that the job ended because of a device error.
*ENDJOBNOLIST	Ends the job when an I/O error occurs. There is no job log produced for the job. The system sends a message to the history log (QHST). This message indicates that the job ended because of a device error.

End severity. The message severity level of escape messages that can cause a batch job to end. The batch job ends when a request in the batch input stream sends an escape message, whose severity is equal to or greater than this value, to the request processing program. The possible values are from 0 through 99.

Hold on job queue. Whether jobs using this job description are put on the job queue in the hold condition. The possible values are *YES and *NO.

»**Initial ASP group information.** The list of initial ASP groups for jobs that use this job description. This does not include the system ASP or basic user ASPs.«

Initial library list. The initial library list that is used for jobs that use this job description. Only the libraries in the user portion of the library list are included.

Note: The data is an array of 11-byte entries, each entry consisting of a 10-byte library name that is left-justified with a blank pad at the end. The 11-byte entries can be easily used in CL commands. The number of libraries in the initial library list tells how many entries are contained in the array.

Inquiry message reply. How inquiry messages are answered for jobs that use this job description.

*RQD	The job requires an answer for any inquiry messages that occur while the job is running.
*DFT	The system uses the default message reply to answer any inquiry messages issued while the job is running. The default reply is either defined in the message description or is the default system reply.
*SYSRPYL	The system reply list is checked to see if there is an entry for an inquiry message issued while the job is running. If a match occurs, the system uses the reply value for that entry. If no entry exists for that message, the system uses an inquiry message.

Job date. The date that will be assigned to jobs using this job description when they are started. The possible values are:

*SYSVAL	The value in the QDATE system value is used at the time the job is started.
job-date	The date to be used at the time the job is started. This date is in the format specified for the DATFMT job attribute.

Job description library name. The name of the library in which the job description resides.

Job description name. The name of the job description about which information is being returned.

Job message queue maximum size. The maximum size (in megabytes) of the job message queue. The possible values are:

0	The maximum size set by system value QJOBMSGMX at the time the job is started.
2-64	The maximum size of the job message queue in megabytes.

Job message queue full action. The action taken when the job message queue becomes full. The possible values are:

SYSVAL The value is specified by the system value QJOBMSGQFL.

NOWRAP When the message queue becomes full, do not wrap. This action will cause the job to end.

WRAP When the message queue becomes full, wrap to the beginning and start filling again.

PRTWRAP When the message queue becomes full, wrap the job queue and print the messages that are being overlaid.

Job queue library name. The library of the job queue into which batch jobs using this job description are placed.

Job queue name. The name of the job queue into which batch jobs using this job description are placed.

Job queue priority. The scheduling priority of each job that uses this job description. The highest priority is 1 and the lowest priority is 9.

Job switches. The initial settings for a group of eight job switches used by jobs that use this job description. These switches can be set or tested in a program and used to control a program's flow. The possible values are '0' (off) and '1' (on).

≫**Length of one initial ASP group information entry.** The length of one initial ASP group information entry. Zero indicates that jobs using this job description do not have an initial ASP group. ≪

Length of request data. The length of all available request data, in bytes. If the receiver variable was not sufficiently large to hold all of the request data available, the amount of request data actually returned may be less than this value.

Logging of CL programs. Whether or not commands are logged for CL programs that are run. The possible values are *YES and *NO.

Message logging level. The type of information logged. Possible types are:

0 No messages are logged.

1 All messages sent to the job's external message queue with a severity greater than or equal to the message logging severity are logged. This includes the indication of job start, job end and job completion status.

2 The following information is logged:

- Level 1 information.
- Request messages that result in a high-level message with a severity code greater than or equal to the logging severity cause the request message and all associated messages to be logged.

  Note: A high-level message is one that is sent to the program message queue of the program that receives the request message. For example, QCMD is an IBM-supplied request processing program that receives request messages.

3 The following information is logged:
- Level 1 and 2 information.
- All request messages.
- Commands run by a CL program are logged if it is allowed by the logging of CL programs job attribute and the log attribute of the CL program.

4 The following information is logged:
- All request messages and all messages with a severity greater than or equal to the message logging severity, including trace messages.
- Commands run by a CL program are logged if it is allowed by the logging of CL programs job attribute and the log attribute of the CL program.

Message logging severity. The severity level that is used in conjunction with the logging level to determine which error messages are logged in the job log. The possible values are from 0 through 99.

Message logging text. The level of message text that is written in the job log when a message is logged according to the logging level and logging severity. The possible values are:

  *MSG   Only the message text is written to the job log.

  *SECLVL   Both the message text and the message help (cause and recovery) of the error message are written to the job log.

  *NOLIST   If the job ends normally, no job log is produced. If the job ends abnormally (if the job end code is 20 or higher), a job log is produced. The messages that appear in the job log contain both the message text and the message help.

≫**Number of initial ASP group information entries.** The number of entries in the job description's initial ASP group information. Zero indicates that jobs using this job description do not have an initial ASP group. ≪

Number of libraries in initial library list. The number of libraries in the user portion of the initial library list.

≫**Offset to initial ASP group information.** The offset in characters (bytes) from the beginning of the structure to the first ASP group information entry. Zero indicates that jobs using this job description do not have an initial ASP group. ≪

Offset to initial library list. The offset from the beginning of the structure to the start of the initial library list.

Offset to request data. The offset from the beginning of the structure to the start of the request data.

Output queue library name. The name of the library in which the output queue resides.

Output queue name. The name of the default output queue that is used for spooled output produced by jobs that use this job description.

  *USRPRF   The output queue name for jobs using this job description is obtained from the user profile of the job at the time the job is started.

  *DEV   The output queue with the same name as the printer device for this job description is used.

*WRKSTN	The output queue name is obtained from the device description from which this job is started.
output-queue-name	The name of the output queue for this job description.

Output queue priority. The output priority for spooled files that are produced by jobs using this job description. The highest priority is 1, and the lowest priority is 9.

Print text. The line of text (if any) that is printed at the bottom of each page of printed output for jobs using this job description. If the special value *SYSVAL is specified, the value in the system value QPRTTXT is used for jobs using this job description.

Printer device name. The name of the printer device or the source for the name of the printer device that is used for all spooled files created by jobs that use this job description.

*USRPRF	The printer device name is obtained from the user profile of the job at the time the job is started.
*SYSVAL	The value in the system value QPRTDEV at the time the job is started is used as the printer device name.
*WRKSTN	The printer device name is obtained from the work station where the job was started.
printer-device-name	The name of the printer device that is used with this job description.

Request data. The request data that is placed as the last entry in the job's message queue for jobs that use this job description. The possible values are:

*NONE	No request data is placed in the job's message queue.
*RTGDTA	The data specified in the routing data parameter is placed as the last entry in the job's message queue.
request-data	The request data to use for jobs that use this job description.

Reserved. An ignored field.

Routing data. The routing data that is used with this job description to start jobs. The possible values are:

QCMDI	The default routing data QCMDI is used by the IBM-supplied interactive subsystem to route the job to the IBM-supplied control language processor QCMD in the QSYS library.
*RQSDTA	Up to the first 80 characters of the request data specified in the request data field are used as the routing data for the job.
routing-data	The routing data to use for jobs that use this job description.

≫**Spooled file action.** Specifies whether spooled files can be accessed through job interfaces once a job has completed its normal activity.

*KEEP	Spooled files are kept with the job when the job completes itsactivity.
*DETACH	Spooled files are detached from the job when the job completes its activity.

SYSVAL The jobs using this job description will take the spooled file action specified by the QSPLFACN system value. «

Syntax check severity. Whether requests placed on the job's message queue are checked for syntax as CL commands, and the message severity that causes a syntax error to end processing of a job. The possible values are:

-1 The request data is not checked for syntax as CL commands. This is equivalent to *NOCHK.

0-99 Specifies the lowest message severity that causes a running job to end. The request data is checked for syntax as CL commands, and, if a syntax error occurs that is greater than or equal to the error message severity specified here, the running of the job that contains the erroneous command is suppressed.

Text description. The user text, if any, used to briefly describe the job description.

Time-slice end pool. Whether interactive jobs using this job description should be moved to another main storage pool when they reach time-slice end. The possible values are:

SYSVAL The system value is used.

NONE The job is not moved when it reaches time-slice end.

BASE The job is moved to the base pool when it reaches time-slice end.

User name. The name of the user profile associated with this job description. If *RQD is specified, a user name is required to use the job description.

Error Messages

Message ID	Error Message Text
CPF1618 E	Job description &1 in library &2 damaged.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9807 E	One or more libraries in library list deleted.

CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Retrieve Job Information (QUSRJOBI) 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)


Optional Parameter Group 1:


    6       Error code                       I/O          Char(*)


Optional Parameter Group 2:


    7       Reset performance statistics     Input        Char(1)


Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes.
```

The Retrieve Job Information (QUSRJOBI) API retrieves specific information about a job.

Authorities and Locks

The following authority restrictions apply only when the API is called for format names JOBI0700, JOBI0750, JOBI0800, JOBI0900. All other format names have no authority restrictions.

Job Authority

> When calling this API for format names JOBI0700, JOBI0750, JOBI0800, and JOBI0900, the API must be called from within the job for which the information is being retrieved or the caller of the API must be running under a user profile that is the same as the job user identity of the job for which the information is being retrieved or the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

> The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the Work Management book on the V5R1 Supplemental Manuals Web site.

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 as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8 bytes.

Format name

> INPUT; CHAR(8)

> The format of the job information to be returned. The format names supported are:

> | | |
> |---|---|
> | *JOBI0100* | Basic performance information |
> | *JOBI0150* | Additional performance information |
> | *JOBI0200* | WRKACTJOB information |
> | *JOBI0300* | Job queue and output queue information |
> | *JOBI0400* | Job attribute information |
> | *JOBI0500* | Message logging information |
> | *JOBI0600* | Active job information |
> | *JOBI0700* | Library list information |
> | *JOBI0750* | Extended library list information |
> | *JOBI0800* | Active job signal information |
> | *JOBI0900* | Active job SQL information |
> | *JOBI1000* | Elapsed performance statistics |

> Refer to [Selecting a Job Information Format](#) for details of each of the formats.

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 that this program is running in. 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 API, QUSLJOB, 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.

Optional Parameter 1

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.

Optional Parameter 2

Reset performance statistics

> INPUT; CHAR(1)
>
> The elapsed time and all fields that are part of the JOBI1000 format, which are based on the elapsed time, will be reset to zero. This field must be zero if other formats are specified. The default value for this field is zero. The following special values may be specified:
>
> *0* The performance statistics will not be reset. The elapsed time will be incremented and the fields in the JOBI1000 format will be recalculated based on the elapsed time interval.
>
> *1* The elapsed time and the fields in the JOBI1000 format will be reset to zero.

Selecting a Job Information Format

All formats may be called against multithreaded jobs; that is, single threaded Job A may retrieve job information about multithreaded Job B. Refer to Considerations for Attribute Scope and Thread Safety for thread safety information when calling these formats from within a multithreaded job.

The following section presents some of the performance characteristics of the different formats (primarily JOBI0100, JOBI0150, and JOBI0200). When formats return some of the same information, the performance effects are discussed. When a format contains information not available in other formats, performance is not discussed.

JOBI0100   This format returns basic performance information about a job. It is faster than the JOBI0150 format and the JOBI0200 format (which also contain performance information). The reason that this format is faster is that it does not touch as many objects, causing less paging when retrieving information about the job.

JOBI0150   This format returns additional performance information, and is slower than the JOBI0100 format. It is similar to the JOBI0200 format, but is faster than that format because there is less paging involved in retrieving the information.

JOBI0200   This format returns information equivalent to that found on the Work with Active Jobs (WRKACTJOB) command.

JOBI0300   This format returns job queue and output queue information for a job, as well as information about the submitter's job if the job is a submitted batch job.

JOBI0400   This format primarily returns job attribute types of information, but has other types of information as well.

JOBI0500   This format returns message logging information.

JOBI0600   This format returns information about active jobs only. It is intended to supplement the JOBI0400 format. It retrieves information from several additional objects associated with the job, and therefore, it causes additional paging.

JOBI0700   This format returns library list information for an active job.

JOBI0750   This format returns library list information for an active job plus additional information about each library returned.

JOBI0800   This format returns signal information for an active job.

JOBI0900   This format returns SQL information for an active job.

JOBI1000   This format returns elapsed performance statistics. Performance values returned are based on an elapsed time (returned as part of this format).

Each format returns information that is only valid for the status of certain jobs. For example, the JOBI0200 format only returns information for active jobs. Because the job status can change between the time the list is generated and the time the Retrieve Job Information API is called, you must design your application to handle this.

When requesting information about a job that has an unknown or incorrect job status for the format requested, the API returns the current status of the job and sets the remainder of the fields for that format to zeros and blanks. When requesting information about a job that is not valid, the API returns the job's status as blanks and sets the remainder of the fields for that format to zeros and blanks. Therefore, you should check the returned status of the job **before** processing the data. Each format description specifies each status for which the API returns complete information.

JOBI0100 Format

The JOBI0100 format information is valid for active jobs and jobs on queues. For jobs on queues, this format returns zeros or blanks for the attributes. If the Change Job (CHGJOB) command was run against a job on a *JOBQ, the attributes returned are the attributes specified on the CHGJOB command. If the job status changes to *OUTQ, the status field returned is *OUTQ and the API returns no information other than the number of bytes returned, the number of bytes available, the qualified job name, the job type, the job

subtype, and the internal job identifier.

The JOBI0100 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	Run priority (job)
68	44	BINARY(4)	Time slice
72	48	BINARY(4)	Default wait
76	4C	CHAR(10)	Purge

JOBI0150 Format

The JOBI0150 format is valid for active jobs only. If the job status changes to *OUTQ or *JOBQ, the status field is set appropriately, and no information other than the number of bytes returned, the number of bytes available, the qualified job name, the job type, the job subtype, and the internal job identifier is returned.

The JOBI0150 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format JOBI0100
86	56	CHAR(10)	Time-slice end pool
96	60	BINARY(4)	Processing unit time used, if less than 2,147,483,647 milliseconds
100	64	BINARY(4)	System pool identifier
104	68	BINARY(4)	Maximum processing unit time
108	6C	BINARY(4)	Temporary storage used in kilobytes
112	70	BINARY(4)	Maximum temporary storage in kilobytes
116	74	BINARY(4)	Thread count
120	78	BINARY(4)	Maximum threads

124	7C	BINARY(4)	Temporary storage used in megabytes
128	80	BINARY(4)	Maximum temporary storage in megabytes
132	84	CHAR(4)	Reserved
136	88	BINARY(8), UNSIGNED	Processing unit time used - total for the job

JOBI0200 Format

The JOBI0200 format is only valid for active jobs and is similar to the information supported by the Work with Active Jobs (WRKACTJOB) command. If the job status has changed to *OUTQ or *JOBQ, the status field is set appropriately, and no information other than the number of bytes returned, the number of bytes available, the qualified job name, the job type, the job subtype, and the internal job identifier is returned.

The JOBI0200 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(10)	Subsystem description name
72	48	BINARY(4)	Run priority (job)
76	4C	BINARY(4)	System pool identifier
80	50	BINARY(4)	Processing unit time used, if less than 2,147,483,647 milliseconds
84	54	BINARY(4)	Number of auxiliary I/O requests, if less than 2,147,483,647
88	58	BINARY(4)	Number of interactive transactions
92	5C	BINARY(4)	Response time total
96	60	CHAR(1)	Function type
97	61	CHAR(10)	Function name
107	6B	CHAR(4)	Active job status
111	6F	BINARY(4)	Number of database lock waits
115	73	BINARY(4)	Number of internal machine lock waits
119	77	BINARY(4)	Number of nondatabase lock waits
124	7C	BINARY(4)	Time spent on database lock waits

Dec	Hex	Type	Field
127	7F	BINARY(4)	Time spent on internal machine lock waits
131	83	BINARY(4)	Time spent on nondatabase lock waits
135	87	CHAR(1)	Reserved
136	88	BINARY(4)	Current system pool identifier
140	8C	BINARY(4)	Thread count
144	90	BINARY(8), UNSIGNED	Processing unit time used - total for the job
152	98	BINARY(8), UNSIGNED	Number of auxiliary I/O requests
160	A0	BINARY(8), UNSIGNED	Processing unit time used for database - total for the job
168	A8	BINARY(8), UNSIGNED	Page faults
176	B0	CHAR(4)	Active job status for jobs ending
180	B4	CHAR(10)	Memory pool name
190	BE	CHAR(1)	Message reply

JOBI0300 Format

This format returns job queue and output queue information for a job, as well as information about the submitter's job. This information is valid for any job status. The JOBI0300 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(10)	Job queue name
72	48	CHAR(10)	Job queue library name
82	52	CHAR(2)	Job queue priority
84	54	CHAR(10)	Output queue name
94	5E	CHAR(10)	Output queue library name
104	68	CHAR(2)	Output queue priority
106	6A	CHAR(10)	Printer device name
116	74	CHAR(10)	Submitter's job name
126	7E	CHAR(10)	Submitter's user name

136	88	CHAR(6)	Submitter's job number
142	8E	CHAR(10)	Submitter's message queue name
152	98	CHAR(10)	Submitter's message queue library name
162	A2	CHAR(10)	Status of job on the job queue
172	AC	CHAR(8)	Date and time job was put on this job queue
180	B4	CHAR(7)	Job date

JOBI0400 Format

This format primarily returns job attribute types of information, but has other types of information as well. This format is valid for any job status. The JOBI0400 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(13)	Date and time job entered system
75	4B	CHAR(13)	Date and time job became active
88	58	CHAR(15)	Job accounting code
103	67	CHAR(10)	Job description name
113	71	CHAR(10)	Job description library name
123	7B	CHAR(24)	Unit of work ID
147	93	CHAR(8)	Mode name
155	9B	CHAR(10)	Inquiry message reply
165	A5	CHAR(10)	Logging of CL programs
175	AF	CHAR(10)	Break message handling
185	B9	CHAR(10)	Status message handling
195	C3	CHAR(13)	Device recovery action
208	D0	CHAR(10)	DDM conversation handling
218	DA	CHAR(1)	Date separator
219	DB	CHAR(4)	Date format
223	DF	CHAR(30)	Print text
253	FD	CHAR(10)	Submitter's job name
263	107	CHAR(10)	Submitter's user name

273	111	CHAR(6)	Submitter's job number
279	117	CHAR(10)	Submitter's message queue name
289	121	CHAR(10)	Submitter's message queue library name
299	12B	CHAR(1)	Time separator
300	12C	BINARY(4)	Coded character set ID
304	130	CHAR(8)	Date and time job is scheduled to run
312	138	CHAR(10)	Print key format
322	142	CHAR(10)	Sort sequence table name
332	14C	CHAR(10)	Sort sequence library
342	156	CHAR(3)	Language ID
345	159	CHAR(2)	Country or region ID
347	15B	CHAR(1)	Completion status
348	15C	CHAR(1)	Signed-on job
349	15D	CHAR(8)	Job switches
357	165	CHAR(10)	Job message queue full action
367	16F	CHAR(1)	Reserved
368	170	BINARY(4)	Job message queue maximum size
372	174	BINARY(4)	Default coded character set identifier
376	178	CHAR(80)	Routing data
456	1C8	CHAR(1)	Decimal format
457	1C9	CHAR(10)	Character identifier control
467	1D3	CHAR(30)	Server type
497	1F1	CHAR(1)	Allow multiple threads
498	1F2	CHAR(1)	Job log pending
499	1F3	CHAR(1)	Reserved
500	1F4	BINARY(4)	Job end reason
504	1F8	BINARY(4)	Job type - enhanced
508	1FC	CHAR(13)	Date and time job ended
≫ 521	209	CHAR(1)	Reserved
522	20A	CHAR(10)	Spooled file action
532	214	BINARY(4)	Offset to ASP group information
536	218	BINARY(4)	Number of entries in ASP group information
540	21C	BINARY(4)	Length of one ASP group information entry
This field repeats for each ASP group information entry.		CHAR(*)	ASP group information entry (See Format of ASP Group Information Entry for more information.)≪

»Format of ASP Group Information Entry

The ASP group information entry describes the data that is returned for each ASP group in the ASP group information of the JOBI0400 format. For details about the fields listed, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
The fields repeat for each ASP group.		CHAR(10)	ASP group name
		CHAR(*)	Reserved«

JOBI0500 Format

This format returns message logging information. This format is valid for any job status. The JOBI0500 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	End severity
68	44	BINARY(4)	Logging severity
72	48	CHAR(1)	Logging level
73	49	CHAR(10)	Logging text

JOBI0600 Format

The JOBI0600 format returns information about active jobs. If the job status changes to *JOBQ or *OUTQ, the status field is set appropriately, and no information other than the number of bytes returned, the number of bytes available, the qualified job name, the job type, the job subtype, and the internal job identifier is returned.

The JOBI0600 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset	

Dec	Hex	Type	Field
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(8)	Job switches
70	46	CHAR(1)	End status
71	47	CHAR(10)	Subsystem description name
81	51	CHAR(10)	Subsystem description library name
91	5B	CHAR(10)	Current user profile
101	65	CHAR(1)	DBCS-capable
102	66	CHAR(1)	Exit key
103	67	CHAR(1)	Cancel key
104	68	BINARY(4)	Product return code
108	6C	BINARY(4)	User return code
112	70	BINARY(4)	Program return code
116	74	CHAR(10)	Special environment
126	7E	CHAR(10)	Device name
136	88	CHAR(10)	Group profile name
146	92	ARRAY(15) of CHAR(10)	Group profile name - supplemental
296	128	CHAR(10)	Job user identity
306	132	CHAR(1)	Job user identity setting
307	133	CHAR(15)	» Client IP address - IPv4 «

JOBI0700 Format

The JOBI0700 format returns library list information for active jobs only. The format returns the actual length instead of the total length because all libraries may not exist. The JOBI0700 format returns the following job information. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name

28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	Number of libraries in SYSLIBL
68	44	BINARY(4)	Number of product libraries
72	48	BINARY(4)	Current library existence
76	4C	BINARY(4)	Number of libraries in USRLIBL
See note	See note	Array(*) of CHAR(11)	System library list
See note	See note	Array(*) of CHAR(11)	Product libraries
See note	See note	Array(*) of CHAR(11)	Current library
See note	See note	Array(*) of CHAR(11)	User library list

Note: The decimal and hexadecimal offsets depend on the number of libraries you have in the various parts of your library lists. The data is left-justified with a blank pad at the end. The array is sequential. It is an array or data structure. See the Control Language information for the total number of libraries that can be returned to you.

JOBI0750 Format

The JOBI0750 format returns library list information for active jobs only along with additional information about each library. The JOBI0750 format returns the following library information for the active job. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	Offset to libraries in system library list
68	44	BINARY(4)	Number of libraries in system library list

72	48	BINARY(4)	Offset to product libraries
76	4C	BINARY(4)	Number of product libraries
80	50	BINARY(4)	Offset to current library
84	54	BINARY(4)	Number of current libraries
88	58	BINARY(4)	Offset to libraries in user library list
92	5C	BINARY(4)	Number of libraries in user library list
96	60	BINARY(4)	Length of one library array entry
See note	See note	Array(*) of CHAR(*)	System library list (See Library array entry for format of library array entry.)
See note	See note	Array(*) of CHAR(*)	Product libraries (See Library array entry for format of library array entry.)
See note	See note	Array(*) of CHAR(*)	Current library (See Library array entry for format of library array entry.)
See note	See note	Array(*) of CHAR(*)	User library list (See Library array entry for format of library array entry.)

Note: The decimal and hexadecimal offsets depend on the number of libraries you have in the various parts of your library lists. The data is left-justified with a blank pad at the end. The array is sequential. It is an array or data structure. See CL Programming book for the total number of libraries that can be returned.

Library array entry

The library array entry describes the data that is returned for each library entry in the array of libraries on the JOBI0750 format. The name of the library as well as some extended information about the library is returned with this format. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
The fields repeat for each library object returned in the array.		CHAR(10)	Library name
		CHAR(50)	Library text description
		» BINARY(4)	Library ASP number
		CHAR(10)	Library ASP name«
		CHAR(*)	Reserved

JOBI0800 Format

The JOBI0800 format is only valid for active jobs. If the job status has changed to *OUTQ or *JOBQ, the status field is set appropriately, and no information other than the number of bytes returned, the number of bytes available, the qualified job name, and the internal job identifier is returned. If the signal status is 0, not enabled for signals, this format returns zeros or blanks for the attributes.

The JOBI0800 format returns the following job information. For details about the fields listed, see Field

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(2)	Reserved
64	40	BINARY(4)	Signal status
68	44	CHAR(8)	Signal blocking mask
76	4C	CHAR(8)	Pending signal set
84	54	BINARY(4)	Offset to signal monitor data
88	58	BINARY(4)	Number of signal monitors
92	5C	BINARY(4)	Process ID number
96	60	Array(*) of CHAR(32)	Signal monitor data (for each signal monitor)
These fields repeat for each signal monitor.		BINARY(4)	Signal number
		BINARY(4)	Signal action
		BINARY(4)	Default signal action
		BINARY(4)	Maximum number of signals retained
		BINARY(4)	Current number of pending signals
		CHAR(12)	Reserved

JOBI0900 Format

The JOBI0900 format is only valid for active jobs. If the job status has changed to *OUTQ or *JOBQ, the status field is set appropriately, and no information other than the number of bytes returned, the number of bytes available, the qualified job name, and the internal job identifier is returned. If the number of SQL open cursors is 0 and no SQL statements have ever been issued in the job, this format returns zeros or blanks for the attributes.

Note: Synchronization is not performed when you change or retrieve SQL data. If you try to retrieve SQL data for your own job and your job is not running multithreaded, then the retrieved data should be correct. If, however, you are retrieving SQL data for your own job and your job is running multithreaded or if you are retrieving data for a different job, the SQL data may not be correct because the SQL data is being changed at the same time that you are retrieving it.

The JOBI0900 format returns the following job information. For details about the fields listed, see [Field Descriptions](#).

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(1)	Server mode for Structured Query Language
63	3F	CHAR(1)	Reserved
64	40	BINARY(4)	Offset to SQL open cursor data
68	44	BINARY(4)	Size of SQL open cursor data
72	48	BINARY(4)	Number of SQL open cursors
76	4C	BINARY(4)	Offset to current SQL statement
80	50	BINARY(4)	Length of current SQL statement
84	54	BINARY(4)	Status of current SQL statement
88	58	BINARY(4)	CCSID of current SQL statement
» 92	5C	CHAR(18)	Relational Database name «
» *	*	Array(*) of CHAR(80)	SQL open cursor data «
These fields repeat for each SQL open cursor		CHAR(10)	Object name for SQL cursor
		CHAR(10)	Object library for SQL cursor
		CHAR(10)	Object type for SQL cursor
		CHAR(18)	SQL cursor name
		CHAR(18)	SQL statement name
See note		CHAR(*)	Current SQL statement
Note: The decimal and hexadecimal offsets depend on the number of SQL open cursors returned in the array. The maximum length of an SQL statement is 32765 bytes.			

JOBI1000 Format

The JOBI1000 format is valid for active jobs only. This format returns performance statistics for the active job based on an elapsed time interval. The first call of this format for the specified job returns zeros for each attribute returned. Upon consecutive calls for the specified job, the values returned for each attribute are calculated based on the time that has elapsed since the last call. The amount of time that has elapsed is returned as part of this format. If the job status has changed to *OUTQ or *JOBQ, the status field is set appropriately, and no information other than the number of bytes returned, the number of bytes available, qualified job name, job type, and the job subtype is returned.

For details about the fields listed, see [Field Descriptions](#).

Offset		Type	Field
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of bytes returned
4	4	BINARY(4)	Number of bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(16)	Internal job identifier
50	32	CHAR(10)	Job status
60	3C	CHAR(1)	Job type
61	3D	CHAR(1)	Job subtype
62	3E	CHAR(2)	Reserved
64	40	BINARY(8), UNSIGNED	Elapsed time
72	48	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time (job)
80	50	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - asynchronous I/O (job)
88	58	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - synchronous I/O (job)
96	60	BINARY(4)	Interactive response time - total during the elapsed time
100	64	BINARY(4)	Interactive transactions - count during the elapsed time
104	68	BINARY(4)	Processing unit used - percent used during the elapsed time (job)
108	6C	BINARY(4)	Processing unit used for database - percent used during the elapsed time (job)
112	70	BINARY(8), UNSIGNED	Processing unit used - time during the elapsed time (job)
120	78	BINARY(8), UNSIGNED	Processing unit used for database - time during the elapsed time (job)
128	80	BINARY(8), UNSIGNED	Lock wait time - time during the elapsed time
136	88	BINARY(8), UNSIGNED	Page fault count during the elapsed time (job)

Field Descriptions

≫Most field descriptions for this API are in [Work Management API Attribute Descriptions (WMAttrDesc)](#). Those field descriptions not found in the Work Management API Attribute Descriptions are listed below.≪

All fields are scoped to the job unless specifically noted. See [Considerations for Attribute Scope and](#)

[Thread Safety](#) for complete details.

»ASP group information entry. Specifies information about an auxiliary storage pool (ASP) group.

ASP group name. The name of the auxiliary storage pool (ASP) group. This is the name of the primary ASP in an ASP group or the name of an ASP device description. The following special values may also be returned:

  *N   The name of the ASP group cannot be determined.

Length of one ASP group information entry. The length of an ASP group information entry. Zero indicates that an ASP group is not being used. Zero is also returned if the job status has changed to *OUTQ.

Length of one library array entry. The length of one entry in a library list array.

Library ASP name. The name of the ASP device that contains the library. The following special values may also be returned:

  *SYSBAS   The library is located in the system ASP or in a basic user ASP.

  *N   The name of the ASP device cannot be determined.

Library ASP number. The numeric identifier of the ASP containing the library. The following values may be returned:

  1   The library is located in the system ASP.

  2-32   The library is located in a basic ASP.

  33-255 The library is located in an independent ASP.

  -1   The ASP number cannot be determined.«

Library name. The name of the library object.

Library text description. The text description of the library object. This field is blank if no text description is specified.

»Number of current libraries. The number of current libraries in the library list of the initial thread.

Number of entries in ASP group information. The number of entries in the ASP group information. Zero indicates that an ASP group is not being used. Zero is also returned if the job status has changed to *OUTQ.

Number of libraries in system library list. The number of libraries in the system part of the library list of the initial thread.

Number of libraries in user library list. The number of libraries in the user library list of the initial thread.

Number of product libraries. The number of product libraries in the library list of the initial thread.

Offset to ASP group information. The offset from the start of the format to the start of the ASP group information. Zero indicates that an ASP group is not being used. Zero is also returned if the job status has changed to *OUTQ.

Offset to current library. The offset from the start of the format to the start of the current library.

Offset to libraries in system library list. The offset from the start of the format to the start of the system library list.

Offset to libraries in user library list. The offset from the start of the format to the start of the user library list.

Offset to product libraries. The offset from the start of the format to the start of the product libraries. «

Comparing Job Type and Subtype with the Work with Active Job Command

The following table compares the job type and job subtype fields returned by the QUSRJOBI API to the type field on the Work with Active Job (WRKACTJOB) command.

WRKACTJOB and QUSRJOBI API Comparison		
Job Type Field	**Job Type**	**Job Subtype**
ASJ (Autostart)	A	blank
BCH (Batch)	B	blank
BCI (Batch immediate)	B	D
EVK (Started by a program start request)	B	E
INT (Interactive)	I	blank
M36 (AS/400 Advanced 36 machine server)	B	F
MRT (Multiple requester terminal)	B	T
PJ (Prestart job)	B	J
PDJ (Print driver job)	W	P
RDR (Reader)	R	blank
SYS (System)	S or X	blank
SBS (Subsystem monitor)	M	blank
WTR (Writer)	W	blank
blank (Alternative user subtype--not an active job)	B	U

Usage Notes

Considerations for Attribute Scope and Thread Safety: This API is primarily intended for retrieving job attributes, but it also retrieves some attributes for the initial thread.

The *Scope* column of [Attribute Scope and Thread Safety](#) table that follows shows whether the attribute is scoped to the job or to the thread. If any attributes currently scoped to the job are moved to the thread level in the future, then this API will be changed to retrieve the value for the initial thread. This API cannot be used to retrieve attributes for a secondary thread.

The *Threadsafe* column of this table indicates whether the attributes are considered to be threadsafe, as well

as indicating whether they are safe under other conditions.

Yes: For this particular API, **Yes** indicates that an attribute can always be retrieved and can be considered correct, which includes thread safety. The API may be called from an initial or secondary thread to retrieve the attributes of the current job or a different job. The job whose attributes are being retrieved may be either single threaded or multithreaded.

Conditional; reason 1, same job: An attribute marked with this value can be safely retrieved from either an initial thread or a secondary thread, but can only be considered to be completely correct when retrieving one's own attribute. When retrieving the attribute from another job, the value retrieved may not be completely correct if the other job is changing the attribute while it is being retrieved.

Conditional; reason 2, initial thread: An attribute marked with this value can only be considered to be completely correct when retrieving one's own attribute and you are running in the initial thread. When retrieving the attribute from another job, you may be running in either an initial thread or a secondary thread, but in either case the value retrieved may not be completely correct if the other job is changing the attribute while it is being retrieved.

Conditional; reason 3, single threaded: An attribute marked with this value can only be considered to be completely correct when retrieving one's own attribute and you are running single threaded. When retrieving the attribute from another job, you may be running in either an initial thread or a secondary thread, but in either case the value retrieved may not be completely correct if the other job is changing the attribute while it is being retrieved.

Conditional; reason 4, active job: An attribute marked with this value can only be considered to be completely correct when retrieving the attribute of an active job. The API may be called from an initial or secondary thread to retrieve the attributes of the current job or a different job. The job whose attributes are being retrieved may be either single threaded or multithreaded. However, if the job whose attribute is being retrieved is on a job queue, the value retrieved may not be completely correct if the attribute is being changed while it is being retrieved.

Conditional; reason 5, not during prestart receive: An attribute marked with this value can be considered to be correct with the following exception. The value may not be completely correct if the attribute is being retrieved for a prestart job while the prestart job is receiving a new request. The API may be called from an initial or secondary thread to retrieve the attributes of the current job or a different job. The job whose attributes are being retrieved may be either single threaded or multithreaded.

No: An attribute marked with this value is not threadsafe, nor is it safe for retrieval when running single threaded. The value retrieved may not be completely correct if the value is being changed while it is being retrieved.

Attribute Scope and Thread Safety		
Attribute	**Scope**	**Threadsafe**
Active job status	Initial thread	No
Active job status for jobs ending	Initial thread	No
Allow multiple threads	Initial thread	Yes
≫ ASP group information entry	Initial thread	Yes≪
Break message handling	Job	Yes
Cancel key	Job	Yes
CCSID of current SQL statement	Job	No
Character identifier control	Job	Yes
≫ Client IP address - IPv4≪	Initial thread	Conditional; reason 2
Coded character set ID	Job	Yes

Completion status	Job	Yes
Country or region ID	Job	Yes
Current library existence	Initial thread	Yes
Current library	Initial thread	Yes
Current number of pending signals	Job	Yes
Current system pool identifier	Initial thread	Yes
Current user profile	Initial thread	Conditional; reason 2
Date and time job became active	Job	Yes
Date and time job ended	Job	Yes
Date and time job entered system	Job	Yes
Date and time job is scheduled to run	Job	Yes
Date and time the job was put on this job queue	Job	No
Date format	Job	Yes
Date separator	Job	Yes
DBCS-capable	Job	Yes
DDM conversation handling	Job	Yes
Decimal format	Job	Yes
Default coded character set identifier	Job	Yes
Default signal action	Job	Yes
Default wait	Job	Yes
Device name	Job	Yes
Device recovery action	Job	Yes
Disk I/O count during the elapsed time - asynchronous I/O (job)	Job	Yes
Disk I/O count during the elapsed time - synchronous I/O (job)	Job	Yes
Disk I/O count during the elapsed time (job)	Job	Yes
Elapsed time	Job	Yes
End severity	Job	Yes
End status	Job	Yes
Exit key	Job	Yes
Function name	Initial thread	No
Function type	Initial thread	No
Group profile name	Initial thread	Conditional; reason 2
Group profile name - supplemental	Initial thread	Conditional; reason 2
Inquiry message reply	Job	Yes
Interactive response time - total during the elapsed time	Job	Yes
Interactive transactions - count during the elapsed time	Job	Yes

Internal job identifier	Job	Yes
Job accounting code	Job	Conditional; reason 1
Job date	Job	Yes
Job description library name	Job	Yes
Job description name	Job	Yes
Job end reason	Job	Yes
Job log pending	Job	Yes
Job message queue full action	Job	Yes
Job message queue maximum size	Job	Yes
Job name	Job	Yes
Job number	Job	Yes
Job queue library name	Job	Conditional; reason 3
Job queue name	Job	Conditional; reason 3
Job queue priority	Job	Yes
Job status	Job	Yes
Job subtype	Job	Yes
Job switches	Job	Conditional; reason 4
Job type	Job	Yes
Job type - enhanced	Job	Yes
Job user identity	Job	Yes
Job user identity setting	Job	Yes
Language ID	Job	Yes
Length of current SQL statement	Job	No
Lock wait time - time during the elapsed time	Job	Yes
Logging level	Job	Yes
Logging of CL programs	Job	Yes
Logging severity	Job	Yes
Logging text	Job	Yes
Maximum number of signals retained	Job	Yes
Maximum processing unit time	Routing step	Yes
Maximum temporary storage in kilobytes	Routing step	Yes
Maximum temporary storage in megabytes	Routing step	Yes
Maximum threads	Job	Yes
Memory pool name	Job	Yes
Message reply	Job	Yes
Mode name	Job	Yes
Number of auxiliary I/O requests	Job	Yes

Number of auxiliary I/O requests, if less than 2,147,483,647	Job	Yes
Number of database lock waits	Initial thread	Yes
Number of interactive transactions	Initial thread	Yes
Number of internal machine lock waits	Initial thread	Yes
Number of libraries in SYSLIBL	Initial thread	Yes
Number of libraries in USRLIBL	Initial thread	Yes
Number of nondatabase lock waits	Initial thread	Yes
Number of product libraries	Initial thread	Yes
Number of signal monitors	Job	Yes
Number of SQL open cursors	Job	No
Object library for SQL cursor	Job	No
Object name for SQL cursor	Job	No
Object type for SQL cursor	Job	No
Offset to current SQL statement	Job	No
Offset to SQL open cursor data	Job	No
Offset to signal monitor data	Job	Yes
Output queue library name	Job	Conditional; reason 3
Output queue name	Job	Conditional; reason 3
Output queue priority	Job	Yes
Page fault count during the elapsed time (job)	Job	Yes
Pending signal set	Job	Yes
Process ID number	Job	Yes
Print key format	Job	Yes
Print text	Job	Conditional; reason 3
Printer device name	Job	Conditional; reason 3
Processing unit time used - total for the job	Job	Yes
Processing unit time used for database - total for the job	Job	Yes
Processing unit time used, if less than 2,147,483,647 milliseconds	Job	Yes
Processing unit used - percent used during the elapsed time (job)	Job	Yes
Processing unit used - time during the elapsed time (job)	Job	Yes
Processing unit used for database - percent used during the elapsed time (job)	Job	Yes
Processing unit used for database - time during the elapsed time (job)	Job	Yes

Product libraries	Initial thread	Yes
Product return code	Job	Yes
Program return code	Job	No
Purge	Job	Yes
» Relational Database name	Job	No«
Response time total	Initial thread	Yes
Run priority (job)	Job	Yes
Server mode for Structured Query Language	Job	No
Server type	Job	Conditional; reason 3
Signal action	Job	Yes
Signal blocking mask	Initial thread	Yes
Signal monitor data	Job	Yes
Signal number	Job	Yes
Signal status	Job	Yes
Signed-on job	Job	Yes
Size of SQL open cursor data	Job	No
Sort sequence	Job	Conditional; reason 3
Sort sequence library	Job	Conditional; reason 3
Special environment	Job	Yes
» Spooled file action	Job	Yes«
SQL cursor name	Job	No
SQL open cursor data	Job	No
SQL statement name	Job	No
Status message handling	Job	Yes
Status of current SQL statement	Job	No
Status of job on the job queue	Job	Yes
Submitter's job name	Job	Yes
Submitter's job number	Job	Yes
Submitter's message queue library name	Job	Yes
Submitter's message queue name	Job	Yes
Submitter's user name	Job	Yes
Subsystem description library name	Job	Yes
Subsystem description name	Job	Yes
System library list	Initial thread	Yes
System pool identifier	Job	Yes
Temporary storage used in kilobytes	Job	Yes
Temporary storage used in megabytes	Job	Yes
Thread count	Job	Yes
Time separator	Job	Yes

Time slice	Job	Conditional; reason 4
Time-slice end pool	Job	Yes
Time spent on database lock waits	Initial thread	Yes
Time spent on internal machine lock waits	Initial thread	Yes
Time spent on nondatabase lock waits	Initial thread	Yes
Unit of work ID	Job	Conditional; reason 5
User library list	Initial thread	Yes
User name	Job	Yes
User return code	Job	Yes

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was 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.
CPF3C54 E	Job &3/&2/&1 currently not available.
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.
CPF9820 E	Not authorized to use library &1.

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

API introduced: V1R3

Retrieve Job Locks (QWCRJBLK) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format of receiver information	Input	Char(8)
4	Job or thread identification information	Input	Char(*)
5	Format of job or thread identification information	Input	Char(8)
6	Error code	I/O	Char(*)

»Optional Parameter Group:

7	Lock filters	Input	Char(*)
8	Format of lock filters	Input	Char(8)«

Default Public Authority: *USE

Threadsafe: Yes

»The Retrieve Job Locks (QWCRJBLK) API generates a list of objects that have been locked or have lower level locks by the job or thread that is specified in the job or thread identification information input parameter.«

Authorities and Locks

Job Authority

> The API must be called from within the job for which the information is being retrieved, or the caller of the API must be running under a user profile that is the same as the job user identity of the job for which the information is being retrieved. Otherwise, the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the [Work Management](#) book on the V5R1 Supplemental Manuals Web site.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a

result, the API returns only the data that the area can hold. For example, this may mean that the number of locked object entries available in the receiver variable doesn't match the value in the number of locked object entries returned.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of the receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of the receiver variable specified is larger then the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of receiver information

> INPUT; CHAR(8)

> The format of the information returned in the receiver variable. The possible format names are:

> | | |
> |---|---|
> | *JBLK0100* | Object level lock format. See JBLK0100 Format for details on the list of objects that this job or thread has locked. |
> | ≫ *JBLK0200* | All object lock format. See JBLK0200 Format for details on the list of objects and members that this job or thread has locked. ≪ |

Job or thread identification information

> INPUT; CHAR(*)

> The information that is used to identify the job or thread for which the job lock information is to be returned. See Format of job or thread identification information for details.

Format of job or thread identification information

> INPUT; CHAR(8)

> The format of the job or thread identification information. The possible format names are:

> | | |
> |---|---|
> | *JIDF0100* | This format is used to retrieve the locks that a job and threads are holding or waiting to hold. See JIDF0100 Format for details on the job or thread identification information. |
> | ≫ *JIDF0200* | This format is used to retrieve the locks that a specific thread is holding or waiting to hold. See JIDF0200 Format for details on the job or thread identification information. ≪ |

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

Lock filters

>INPUT;CHAR(*)

>Filters used for the lock information that is returned. See the Lock filter format for further information.

Format of lock filters

>INPUT; CHAR(8)

>The format of the lock filters used on the returned data. The possible format name is:

>>*JBFL0100*   Lock filter format. See JBFL0100 Format for details on the filters contained in this format.≪

JBLK0100 Format

This format is used to return only OS/400 external objects that are locked.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of locked object entries available
12	C	BINARY(4)	Offset to list of locked objects
16	10	BINARY(4)	Number of locked object entries returned
20	14	BINARY(4)	Length of locked object entry
24	18	*	Reserved
These fields repeat, in the order listed.		CHAR(10)	Object name
		CHAR(10)	Object library name
		CHAR(10)	Object type
		CHAR(10)	Extended object attributes
		CHAR(10)	Lock state
		CHAR(2)	Reserved
		BINARY(4)	Lock status
		BINARY(4)	Member locks
		BINARY(4)	Lock count
		CHAR(1)	Lock scope
		CHAR(3)	Reserved
		CHAR(8)	Thread identifier
		≫BINARY(4) UNSIGNED	Thread handle
		CHAR(20)	Lock space identifier
		CHAR(10)	Object ASP name

	CHAR(10)	Object library ASP name
	BINARY(4)	Object ASP number
	BINARY(4)	Object library ASP number《

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. Only complete entries are returned.

Extended object attributes. The extended attributes of an object. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG (RPG program) or CLP (CL program). 》This field will be blank if there is no extended attribute associated with the object type.《

Length of locked object entry. The length of each locked object entry.

Lock count. The number of identical locks on this entity.

》Lock scope. The scope of the lock. The lock may be a job scope lock, a thread scope lock, or a lock space scope lock. Lower level locks are returned and can occur when a member of a file is locked, but the file itself is not locked. The possible values are:

Blank   The object is not locked, but there are locks on lower level objects.《

0   Job scope.

1   Thread scope.

》 *2*   Lock space scope.

Lock space identifier. When the lock scope field indicates a lock space scope lock, this field contains the identifier of the lock space for which the lock is being waited on. Otherwise, this field is blank.《

Lock state. The lock condition for the lock request. Lower level locks are returned and can occur when a member of a file is locked but the file itself is not locked. The possible values are:

》 *Blank*   The object is not locked but there are locks on lower level objects.《

*SHRRD   Lock shared for read.

*SHRUPD   Lock shared for update.

*SHRNUP   Lock shared, no update.

*EXCLRD   Lock exclusive, read allowed.

*EXCL   Lock exclusive, no read allowed.

Lock status. The status of the lock request. Lower level locks are returned and can occur when a member of a file is locked but the file itself is not locked. Possible values are:

》 *0*   The object is not locked but there are locks on lower level objects.《

1   The lock on this object currently is held by the job or thread.

2   The job or thread is waiting to get the lock on this object (synchronous).

3   The job or thread has a lock request outstanding for this object (asynchronous). The lock may be a single request or part of a multiple lock request for which some other object specified in the request has been identified as unavailable.

Member locks. The number of member locks for a database file. If the object is not a database file, 0 is returned.

Number of locked object entries available. The number of locked object entries that are held by this job and specified threads.

Number of locked object entries returned. The number of locked object entries that are returned.

≫**Object ASP name.** The name of the auxiliary storage pool (ASP) containing the object that is locked.

The following special values also may be returned:

*SYSBAS   The object is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Object ASP number. The numeric identifier of the ASP containing the locked object. The following values may be returned:

1   The object is located in the system ASP.

2-32   The object is located in a basic user ASP.

33-255 The object is located in an independent ASP.

-1   The ASP number cannot be determined.

Object library ASP name. The name of the ASP containing the library of the locked object.

The following specials value also may be returned:

*SYSBAS   The object's library is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Object library ASP number. The numeric identifier of the ASP containing the library of the locked object. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255 The library is located in an independent ASP.

-1   The ASP number cannot be determined.≪

Object library name. The name of the library containing the locked object.

≫The following special value also may be returned:

N   The name of the library cannot be determined.≪

Object name. The name of the object that is locked.

≫The following special value also may be returned:

N   The name of the object cannot be determined.≪

Object type. The object type. For a list of all the available external OS/400 object types, see the Control Language (CL) topic.

Offset to list of locked objects. The offset in bytes from the beginning of the receiver variable to the first entry.

Reserved. An unused field.

≫**Thread handle.** A value that addresses a particular thread within a job holding a thread scope lock or the thread waiting for a lock; otherwise, this is zero. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread.≪

Thread identifier. A value that uniquely identifies a thread within a job holding a thread scope lock or the thread waiting for a lock; otherwise, hexadecimal zeros are returned.

≫JBLK0200 Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of locked object entries available
12	C	BINARY(4)	Offset to list of locked objects
16	10	BINARY(4)	Number of locked object entries returned
20	14	BINARY(4)	Length of locked object entry
24	18	*	Reserved
These fields repeat, in the order listed.		BINARY(4)	Type of entity
		CHAR(30)	Extended object name
		CHAR(10)	Object library name
		CHAR(10)	Object ASP name
		CHAR(10)	Object library ASP name
		BINARY(4)	Object ASP number
		BINARY(4)	Object library ASP number
		CHAR(10)	Object type
		CHAR(10)	Extended object attributes
		CHAR(10)	Member name
		CHAR(1)	Member lock type

	CHAR(3)	Reserved
	CHAR(10)	Lock state
	BINARY(4)	Lock status
	BINARY(4)	Member locks
	BINARY(4)	Lock count
	CHAR(1)	Lock scope
	CHAR(3)	Reserved
	BINARY(8) UNSIGNED	Space location lock offset
	CHAR(8)	Thread identifier
	BINARY(4) UNSIGNED	Thread handle
	CHAR(20)	Lock space identifier
	CHAR(64)	Object lock handle
	CHAR(64)	Lock request handle

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. Only complete entries are returned.

Extended object attributes. The extended attributes of an object. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG (RPG program) or CLP (CL program). This field will be blank if there is no extended attribute associated with the object type.

Extended object name. The name of the object that is locked. This field will be blank if the name is for an internal system object or an internal system object space location, and the user does not have *JOBCTL special authority. If the lock is on a database member then the object name will be the name of the file that owns the member. If the member lock type is member or access path, then the file that owns the member may be either a physical file or a logical file. If the member lock type is data, then the file that owns the member will be a physical file. If the lock is on a lock space then the name will be the lock space id for that lock space.

The following special value also may be returned:

  *N   The name of the object cannot be determined.

Length of locked object entry. The length of each locked object entry.

Lock count. The number of identical locks on this entity.

Lock request handle. A handle to lock request information. Using the Retrieve Lock Request Information (QWCRLRQI) API and passing in this handle you can retrieve additional information about the program that requested this lock. A value of hexadecimal zero is returned when additional information cannot be retrieved. This value is a temporary value that can expire. See the QWCRLRQI API for additional information.

Lock scope. The scope of the lock. The lock may be a job scope lock, a thread scope lock, or a lock space scope lock. Lower level locks are returned and can occur when a member of a file is locked but the file itself is not locked. The possible values are:

Blank   The object is not locked but there are locks on lower level objects.

0   Job scope.

1   Thread scope.

2   Lock space scope.

Lock space identifier. When the lock scope field indicates a lock space scope lock, this field will contain the identifier of the lock space for which the lock is being waited on. Otherwise, this field is blank.

Lock state. The lock condition for the lock request. Lower level locks are returned and can occur when a member of a file is locked but the file itself is not locked. Possible other values are:

Blank   The object is not locked, but there are locks on lower level objects.

*SHRRD   Lock shared for read.

*SHRUPD   Lock shared for update.

*SHRNUP   Lock shared, no update.

*EXCLRD   Lock exclusive, read allowed.

*EXCL   Lock exclusive, no read allowed.

Lock status. The status of the lock request. Possible values are:

0   The object is not locked but there are locks on lower level objects.

1   The lock on this object currently is held by the job or thread.

2   The job or thread is waiting to get the lock on this object (synchronous).

3   The job or thread has a lock request outstanding for this object (asynchronous). The lock may be a single request or part of a multiple lock request for which some other object specified in the request has been identified as unavailable.

Member locks. The number of member locks for a database file. If the object is not a database file, 0 is returned.

Member lock type. If the lock is on a member then this field indicates the type of member lock, otherwise it will be blank. Possible values are:

Blank   The object type is not a member

0   The lock is a member lock

1   The lock is a data lock.

2   The lock is an access path lock.

Member name. The name of the member that has a lock held or waiting on it. If the type of entity is not a member object then this field is blank. The following special value also can be returned:

N   The name of the object cannot be determined.

Number of locked object entries available. The number of locked object entries that are held by this job and specified threads.

Number of locked object entries returned. The number of locked object entries that are returned.

Object ASP name. The name of the Auxiliary Storage Pool (ASP) that contains the object that is locked.

The following special values also may be returned:

*SYSBAS   The object is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Object ASP number. The numeric identifier of the ASP containing the locked object. The following values may be returned:

1   The object is located in the system ASP.

2-32   The object is located in a basic user ASP.

33-255   The object is located in an independent ASP.

-1   The ASP number cannot be determined.

Object library ASP name. The name of the ASP containing the library of the locked object.

The following specials value also may be returned:

*SYSBAS   The object's library is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Object library ASP number. The numeric identifier of the ASP containing the library of the locked object. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP number cannot be determined.

Object library name. The name of the library containing the locked object. This field will be blank if the type of entity is an internal system object or is an internal system object space location.

The following special value also may be returned:

*N   The name of the library cannot be determined.

Object lock handle. An identifier that can be input to Retrieve Lock Information (QWCRLCKI) API to find additional information about other holders of locks on this object. Hexadecimal zeros are returned when additional information cannot be retrieved. The object lock handle is a temporary value that can expire. See the QWCRLCKI API for additional information.

Object type. The object type. For a list of all the available external OS/400 object types, see the Control Language (CL) topic. For a list of all internal system object types, see the OS/400 Diagnostic Aids. Note if the lock is on a database member the object type will be *FILE.

Offset to list of locked objects. The offset in bytes from the beginning of the receiver variable to the first entry.

Reserved. An unused field.

Space location lock offset. A value in bytes to the location in the space that is locked. For objects that are not space location locks this value will be zero.

Thread handle. A value which addresses a particular thread within a job holding a thread scope lock or the thread waiting for a lock, otherwise this is zero. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread.

Thread identifier. A value which uniquely identifies a thread within a job holding a thread scope lock or the thread waiting for a lock; otherwise, hexadecimal zeros are returned.

Type of entity. This is a value that will identify the type of entity for which the lock information is returned.

The following values may be returned:

1	OS/400 external object
2	Member object
3	Internal system object
4	OS/400 external object space location
5	Internal system object space location
6	Lock space object
999	Unknown type

Lock filter format

The format of the lock filter used on the returned lock information.

JBFL0100 Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Filter size
4	4	BINARY(4)	Filter lock state
8	8	BINARY(4)	Filter lock scope
12	C	BINARY(4)	Filter lock status

16	10	CHAR(1)	Include OS/400 external objects flag
17	11	CHAR(1)	Include member objects flag
18	12	CHAR(1)	Include internal system objects flag
19	13	CHAR(1)	Include OS/400 external object space locations flag
20	14	CHAR(1)	Include internal system object space locations flag
21	15	CHAR(1)	Include lock space objects flag
22	16	CHAR(1)	Include unknown entities flag
23	17	CHAR(10)	Filter object name
33	21	CHAR(10)	Filter object library name
43	2B	CHAR(10)	Filter object library ASP name

Field Descriptions

Filter lock scope: This value is used to filter information that is returned so that it contains only information about locks that have a certain lock scope.

0   Do not filter on lock scope

1   Return only the job scope locks

2   Return only the thread scope locks

3   Return only the lock space scope locks

Default   Do not filter on lock scope.

Filter lock state: This value is used to filter information that is returned so that it contains only information about locks that have a certain lock state.

0   Do not filter on lock state

1   Return only the shared locks

2   Return only the exclusive locks

Default   Do not filter on lock state.

Filter lock status: This value is used to filter information that is returned so that it contains only information about locks that have a certain lock status.

0   Do not filter on lock status

1   Return only locks with a status of held

2   Return only locks with a status of waiting

3   Return only locks with a status of requested.

Default   Do not filter on lock status.

Filter object library ASP name: The name of the library's Auxiliary Storage Pool (ASP) to be filtered on. Special value of *SYSBAS can be specified. A blank field will cause no filtering to be done on this field. The default is not to filter on this field.

Filter object library name: This is the library name to be filtered on. A blank field will cause no filtering to be done on this field. The default is not to filter on this field.

Filter object name: Only locks on the specified object will be returned. In the case of database files, locks on members of the file may also be returned. A blank field will cause no filtering to be done on this field. The default is not to filter on this field.

Filter size: The size of the filter information passed. Valid values are:

4   No filtering will be performed. The default values will be used for each filter.

53   All filters will be required

Include internal system objects flag: A value of 1 in this field allows internal system object locks to be returned. A value of 0 will cause these values to be excluded from the return data. Note this field is only valid for the JBLK0200 format. The default is to exclude internal system objects.

Include internal system object space locations flag: A value of 1 in this field allows internal system object space location locks to be returned A value of 0 will cause these values to be excluded from the return data. Note this field is only valid for the JBLK0200 format. The default is to exclude internal system object space locations.

Include lock space objects flag: A value of 1 in this field will allow lock space objects locks to be returned. A value of 0 will cause these values to be excluded from the return data. Note this field is only valid for the JBLK0200 format. The default is to include lock space objects.

Include member objects flag: A value of 1 in this field will allow member objects locks to be returned. A value of 0 will cause these values to be excluded from the return data. Note this field is only valid for the JBLK0200 format. The default is to include member objects.

Include OS/400 external objects flag: A value of 1 in this field will allow OS/400 external object locks to be returned. A value of 0 will cause these objects to be excluded from the return data. Note this field is only valid for the JBLK0200 format. The default is to include OS/400 external objects.

Include OS/400 external object space locations flag: A value of 1 in this field will allow OS/400 external space locations locks to be returned. A value of 0 will cause these values to be excluded from the return data. Note this field is only valid for the JBLK0200 format. The default is exclude OS/400 external object space locations.

Include Unknown types flag: A value of 1 in this field allows locks that are of an unknown entity type to be returned. A value of 0 causes these values to be excluded from the return data. This field is valid for the JBLK0200 format only. The default is exclude unknown types.≪

Format of job or thread identification information

The format of the information needed to identify the job or thread whose locked object information is returned.

JIDF0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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 the job name.

Job name. A specific job name or one of the following special values:

*	The job in which this program is running. The job number and user name must contain blanks.

*INT	The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread identifier. A value that uniquely identifies a thread within a job. A valid thread identifier must be specified.

Thread indicator. The value that is used to specify the thread within the job for which information is to be retrieved. The following values are supported:

0	Information should be retrieved for the thread specified in the thread identifier field.

1	Information should be retrieved for the thread in which this program is running currently.

2	Information should be retrieved for the initial thread of the identified job.

3	Information should be retrieved for the job and its associated threads.

Note: For all supported values, the combination of the internal job identifier, job name, job number, and user name fields must identify the job containing the thread or threads.

User name. A specific user profile name, or blanks when the job name specified is a special value.

»JIDF0200 Format.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4), UNSIGNED	Thread handle
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

*   The job in which this program is running. The job number and user name must contain blanks.

INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread handle. An unused field on this API. This field must contain hex zeros.

Thread identifier. A value that uniquely identifies a thread within a job. A valid thread identifier must be specified.

User name. A specific user profile name, or blanks when the job name specified is a special value.«

Error Messages

Message ID	Error Message Text
CPF136A E	Job &3/&2/&1 not active.
CPF18BF E	Thread &1 not found.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C3B E	Value for parameter &2 for API &1 not valid.

CPF3C3C E   Value for parameter &1 not valid.

CPF3CF1 E   Error code parameter not valid.

CPF3CF2 E   Error(s) occurred during running of &1 API.

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.

CPF3C36 E   Number of parameters, &1, entered for this API was 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.

API introduced: V5R1

Retrieve Job Queue Information (QSPRJOBQ) 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 queue name       Input      Char(20)
    5    Error Code                     I/O        Char(*)


Default Public Authority: *USE

Threadsafe: No
```

The Retrieve Job Queue Information (QSPRJOBQ) API retrieves information associated with a specified job queue.

Authorities and Locks

Job Queue Library Authority

   The caller needs *EXECUTE authority to the job queue library.

Job Queue Authority

   The caller needs one of the following:

   ❍ *READ authority to the job queue.
   ❍ Job control special authority (*JOBCTL) if the job queue is operator controlled
     (OPRCTL(*YES)).
   ❍ Spool control special authority (*SPLCTL).

Job Queue Lock

   This API gets an *EXCLRD lock on the job queue.

Subsystem Description Lock

   This API gets an *EXCLRD lock on the subsystem description.

This API does not check the caller's authority to the subsystem description or subsystem description library when retrieving the subsystem description information.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is to receive the information requested.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided by the receiver variable parameter. The amount of data returned is truncated if it is too small. A length of less than 8 is not valid.

Format name

> INPUT; CHAR(8)

> The content and format of the queue information being returned. The valid format names are:

> | | |
> |---|---|
> | *JOBQ0100* | Basic job queue information. See [JOBQ0100 Format](#) to view the information returned for this format. |
> | *JOBQ0200* | Detailed job queue information. See [JOBQ0200 Format](#) to view the information returned for this format. |

Qualified job queue name

> INPUT; CHAR(20)

> The name of the job queue for which information is returned. The first 10 characters contain the queue name, and the second 10 characters contain the name of the library in which the queue resides.

> The following special values are supported for the library name:

> | | |
> |---|---|
> | *LIBL* | The library list used to locate the job queue. |
> | *CURLIB* | The current library for the job is used to locate the job queue. |

Error code

> I/O; CHAR(*)

> The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

JOBQ0100 Format

The following table shows the information returned for the JOBQ0100 format. For more details about the fields in the following table, see [Field Descriptions](#).

Offset		

Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Job queue name
18	12	CHAR(10)	Job queue library name
28	1C	CHAR(10)	Operator controlled
38	26	CHAR(10)	Authority to check
48	30	BINARY(4)	Number of jobs
52	34	CHAR(10)	Job queue status
62	3E	CHAR(10)	Subsystem name
72	48	CHAR(50)	Text description
122	7A	CHAR(10)	Subsystem library name
132	84	BINARY(4)	Sequence number
136	88	BINARY(4)	Maximum active
140	8C	BINARY(4)	Current active

JOBQ0200 Format

The following table shows the information returned for the JOBQ0200 format. For more details about the fields in the following table see, Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Job queue name
18	12	CHAR(10)	Job queue library name
28	1C	CHAR(10)	Operator controlled
38	26	CHAR(10)	Authority to check
48	30	BINARY(4)	Number of jobs
52	34	CHAR(10)	Job queue status
62	3E	CHAR(10)	Subsystem name
72	48	CHAR(10)	Subsystem library name
82	52	CHAR(50)	Text description
132	84	BINARY(4)	Sequence Number
136	88	BINARY(4)	Maximum active
140	8C	BINARY(4)	Current active
144	90	BINARY(4)	Maximum active jobs with priority 1
148	94	BINARY(4)	Maximum active jobs with priority 2
152	98	BINARY(4)	Maximum active jobs with priority 3
156	9C	BINARY(4)	Maximum active jobs with priority 4
160	A0	BINARY(4)	Maximum active jobs with priority 5

164	A4	BINARY(4)	Maximum active jobs with priority 6
168	A8	BINARY(4)	Maximum active jobs with priority 7
172	AC	BINARY(4)	Maximum active jobs with priority 8
176	B0	BINARY(4)	Maximum active jobs with priority 9
180	B4	BINARY(4)	Active jobs with priority 0
184	B8	BINARY(4)	Active jobs with priority 1
188	BC	BINARY(4)	Active jobs with priority 2
192	C0	BINARY(4)	Active jobs with priority 3
196	C4	BINARY(4)	Active jobs with priority 4
200	C8	BINARY(4)	Active jobs with priority 5
204	CC	BINARY(4)	Active jobs with priority 6
208	D0	BINARY(4)	Active jobs with priority 7
212	D4	BINARY(4)	Active jobs with priority 8
216	D8	BINARY(4)	Active jobs with priority 9
220	DC	BINARY(4)	Released jobs on queue with priority 0
224	E0	BINARY(4)	Released jobs on queue with priority 1
228	E4	BINARY(4)	Released jobs on queue with priority 2
232	E8	BINARY(4)	Released jobs on queue with priority 3
236	EC	BINARY(4)	Released jobs on queue with priority 4
240	F0	BINARY(4)	Released jobs on queue with priority 5
244	F4	BINARY(4)	Released jobs on queue with priority 6
248	F8	BINARY(4)	Released jobs on queue with priority 7
252	FC	BINARY(4)	Released jobs on queue with priority 8
256	100	BINARY(4)	Released jobs on queue with priority 9
260	104	BINARY(4)	Scheduled jobs on queue with priority 0
264	108	BINARY(4)	Scheduled jobs on queue with priority 1
268	10C	BINARY(4)	Scheduled jobs on queue with priority 2
272	110	BINARY(4)	Scheduled jobs on queue with priority 3
276	114	BINARY(4)	Scheduled jobs on queue with priority 4
280	118	BINARY(4)	Scheduled jobs on queue with priority 5
284	11C	BINARY(4)	Scheduled jobs on queue with priority 6
288	120	BINARY(4)	Scheduled jobs on queue with priority 7
292	124	BINARY(4)	Scheduled jobs on queue with priority 8
296	128	BINARY(4)	Scheduled jobs on queue with priority 9
300	12C	BINARY(4)	Held jobs on queue with priority 0
304	130	BINARY(4)	Held jobs on queue with priority 1
308	134	BINARY(4)	Held jobs on queue with priority 2
312	138	BINARY(4)	Held jobs on queue with priority 3
316	13C	BINARY(4)	Held jobs on queue with priority 4
320	140	BINARY(4)	Held jobs on queue with priority 5
324	144	BINARY(4)	Held jobs on queue with priority 6
328	148	BINARY(4)	Held jobs on queue with priority 7

| 332 | 14C | BINARY(4) | Held jobs on queue with priority 8 |
| 336 | 150 | BINARY(4) | Held jobs on queue with priority 9 |

Field Descriptions

Active jobs with priority 0 through 9. The number of jobs that are active for each priority level (0 through 9). If the subsystem name and subsystem library name are blank, then this field is 0.

Authority to check. Whether the user must be the owner of the queue in order to control the queue by holding or releasing the queue. The possible values are:

OWNER   Only the owner of the job queue can control the queue.

DTAAUT   Any user with *READ, *ADD, or *DELETE authority to the job queue can control the queue.

Bytes available. Total format data length.

Bytes returned. Length of the data returned.

Job queue library name. The name of the library that contains the job queue.

Job queue name. The name of the job queue.

Job queue status. The status of the job queue. The status may be one of the following values:

RELEASED   The queue is released.

HELD   The queue is held.

Maximum active. The maximum number of jobs that can be active at the same time through this job queue entry. A -1 in this field indicates that the value is *NOMAX.

Maximum active jobs with priority 1 through 9. The maximum number of jobs that can be active at the same time for each priority level (1 through 9). A -1 in this field indicates that the value is *NOMAX. If the subsystem name and subsystem library name are blank, then this field is 0.

Number of jobs. The number of jobs in the queue.

Operator controlled. Whether a user who has job control authority is allowed to control this job queue and manage the jobs on the queue.

YES   Users with job control authority can control the queue and manage the jobs on the queue.

NO   This queue and its jobs cannot be controlled by users with job control authority unless they also have other special authority.

Released jobs on queue with priority 0 through 9. The number of jobs currently sitting on the job queue in *RELEASED status for each priority level (0 through 9).

Scheduled jobs on queue with priority 0 through 9. The number of jobs currently sitting on the job queue in *SCHEDULED status for each priority level (0 through 9).

Sequence number. The job queue entry sequence number. The subsystem uses this number to determine the order in which job queues are processed. Jobs from the queue with the lowest sequence number are processed first.

Subsystem name. The name of the subsystem that can receive jobs from this job queue. If there is no name, then this queue is not associated with an active subsystem, and no job can be processed.

Subsystem library name. The library in which the subsystem description resides. If there is no name, then this queue is not associated with with an active subsystem and no job can be processed.

Text description. Text that briefly describes the job queue.

   BLANK   There is no text description of the job queue.

Error Messages

Message ID	Error Message Text
CPF1608 E	Subsystem description &1 not found.
CPF2207 E	Not authorized to use object &1 in library &3 type *&2.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3307 E	Job queue &1 in &2 not found.
CPF3330 E	Necessary resource not available.
CPF8121 E	&8 damage on job queue &4 in library &9.
CPF8122 E	&8 damage on library &4.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Retrieve Job Status (QWCRJBST) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Job identifier	Input	Char(*)
4	Format of job identifier	Input	Char(8)
5	Error Code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Retrieve Job Status (QWCRJBST) API returns status and job identification information about the job that is identified by the job identifier parameter. The QWCRJBST API retrieves this information faster than other APIs. It should be considered for use in performance critical applications where the returned information is required.

Authorities and Locks

None.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

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

Length of receiver variable

INPUT BINARY(4)

The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Job identifier

INPUT CHAR(*)

The identifier of the job for which the information is to be retrieved. The job can be identified in one of three ways: job number only, internal job number, or fully qualified job name. The next parameter specifies which format of job identifier is being used.

Format of job identifier

    INPUT CHAR(8)

    The format of the job identifier being provided. The format names that can be used are as follows:

    JOBS0100   The job identifier is a 6-character job number. It is possible that more than one job may have the same job number. This API returns the requested information for only the first job that has the specified job number. No indication is returned to show if more than one job has the same job number.

    JOBS0200   The job identifier is a 16-character internal job number. The internal job number is obtained through the List Job (QUSLJOB) API or as output to this API.

    JOBS0300   The job identifier is a 26-character fully qualified 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.

Format of Returned Information

The information returned from this API has the following format:

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	CHAR(10)	Job status
18	12	CHAR(16)	Internal job identifier
34	22	CHAR(26)	Fully qualified job name

Field Description

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.

Fully qualified job name. The fully qualified job name consisting of three parts. The first 10 characters contain the job name. The next 10 characters contain the user name. The last 6 characters contain the job number.

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

Job status. Possible values that can be returned for job status are as follows:

ACTIVE The job has started, and it can use system resources (processing unit, main storage, and so on). This does not guarantee that the job is currently running, however. For example, an active job may be in one of the following states where it cannot use system resources:

- The Hold Job (HLDJOB) command holds the job; the Release Job (RLSJOB) command allows the job to run again.

- The Transfer Group Job (TFRGRPJOB) command or the Transfer Secondary Job (TFRSECJOB) command suspends the job. When control returns to the job, the job can run again.

- The job is disconnected using the Disconnect Job (DSCJOB) command. When the interactive user signs back on, thereby connecting back into the job, the job can run again.

- The job is waiting for any reason. For example, with an inquiry message, the job can start running again when it receives the reply.

JOBQ The job currently is on a job queue. The job may have been previously active and was placed back on the job queue because of the Transfer Job (TFRJOB) command or the Transfer Batch Job (TFRBCHJOB) command, or the job was never active because it was just submitted.

OUTQ The job has completed running and has spooled output that has not yet printed.

ERROR Either a job with the specified job identifier does not exist, or an error was encountered while attempting to determine its status.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3C51 E	Internal job identifier not valid.
CPF3C52 E	Internal job identifier no longer valid.
CPF3C53 E	Job &3/&2/&1 not found.

CPF3C54 E   Job &3/&2/&1 currently not available.

CPF3C55 E   Job &3/&2/&1 does not exist.

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

API introduced: V3R6

»Retrieve Lock Information (QWCRLCKI) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Object identification	Input	Char(*)
5	Object identification format	Input	Char(8)
6	Number of key fields to be returned	Input	Binary(4)
7	Key of fields to be returned	Input	Array(*) of Binary(4)
8	Filters	Input	Char(*)
9	Filter format	Input	Char(8)
10	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Lock Information (QWCRLCKI) API generates a list of information about lock holders of the item specified.

Authorities and Locks

Object Library Authority

> *EXECUTE

Object Library ASP Device Authority

> *EXECUTE

Note: If the user does not have *EXECUTE authority to the object's library and *EXECUTE authority to the object library's ASP device, the user must have *JOBCTL authority.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold. For example, this may mean that the number of lock information entries returned in the receiver variable doesn't match the value in the number of lock information entries available.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable provided. The length of the receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of the receiver variable specified is larger then the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The name of the format used to retrieve lock information. You can specify this format:

LCKI0100   Lock Information Format. See [LCKI0100 Format](#) for more information.

Object identification

INPUT; CHAR(*)

The information that is used to identify the entity that may be locked. See [Format of Object Identification](#) for more information.

Object identification format

INPUT; CHAR(8)

The format of the entity identification information. Possible format names are:

LOBJ0100   Object name. See [LOBJ0100 Format](#) for more information on this format.

LOBJ0200   Object lock handle. See [LOBJ0200 Format](#) for more information on this format.

Number of key fields to be returned

INPUT; BINARY(4)

The number of key fields to be returned in the LCKI0100 format. If the lock holder type returned is a lock space then the values returned in the key fields will be blank or zero.

Key of fields to be returned.

INPUT; ARRAY(*) of BINARY(4)

The list of fields returned in the LCKI0100 format. For a list of valid fields, see [Valid Keys](#).

Filters

INPUT;CHAR(*)

Filters used for the lock information that is returned. See the [Filter Format](#) for further information.

Filter format

INPUT; CHAR(8)

The format of the lock filter used on the returned data. The possible format name is:

LKFL0100   See [LKFL0100 Format](#) for details on the filters contained in this format.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Format of Object Identification

The formats for the object identification are specified.

LOBJ0100 Format

An external object is specified in this format. See the field descriptions for restrictions that may exist.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Object identification size
4	4	CHAR(10)	Object name specified
14	E	CHAR(10)	Object library name specified
24	18	CHAR(10)	Object library ASP name specified
34	22	CHAR(10)	Object type specified
44	2C	CHAR(10)	Member name
54	36	CHAR(2)	Reserved
56	38	BINARY(4)	Record lock indicator
60	3C	UNSIGNED BINARY(4)	Relative record number

Field Descriptions

Member name. Lock information is retrieved on the specified member of a database file. This is valid only when a database file has been specified for object parameters. For other than database files, use *NONE. Special values used are:

*NONE   No member locks are retrieved, but file level locks are retrieved. If the qualified object name is not a database file, use this value.

Object identification size. The amount of data provided for the LOBJ0100 format. This field must be set to 64.

Object library ASP name specified. The name of the auxiliary storage pool (ASP) device that contains the object's library. This parameter must be * if the library portion of the qualified object name is *CURLIB or *LIBL. Special values used are:

*   The current thread's library name space will be searched.

*SYSBAS   The system ASP and defined basic user ASPs will be searched.

Object library name specified. The name of the library where the object is located. You can use these special values for the library name:

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

*LIBL   The library list is used to locate the object.

Object name specified. The name of the external OS/400 object whose lock information entries are to be placed in the list.

Object type specified. The type of OS/400 object. For a list of all the available external OS/400 object types, see the [Control Language (CL)](#) topic.

Record lock indicator. The indicator that controls the retrieval of record locks that may exist if the object specified is a file and member. If the object specified is not a database file and member or the special value of *ALL* is specified for member, then the value 0 must be used for this field. One of the following values are required for this field.

0   No record lock information is retrieved.

1   The record lock information in the specified file and member will be returned.

Relative record number. The record number in the specified database file and member for which lock information is to be returned. If the object specified is not a database file and member, the value 0 must be used for this field. The following special value is allowed:

0   Record lock information for all records in the member should be returned.

Reserved. This field must be set to hexadecimal zeros.

LOBJ0200 Format

This format is used to find locks where a handle was passed that has been returned in a previous lock call to the Retrieve Job Locks [(QWCRJBLK)](#) API. This call must have been made in the same thread for the handle to be valid. The following fields are used if LOBJ0200 input format is specified.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Object handle size
4	4	CHAR(64)	Object lock handle

Field Descriptions

Object lock handle. A value that identifies the object that is stored from a previous API call. The handle is a temporary value. A storage limitation will allow a thread to create up to 1 million valid handles. Once the storage limit has been reached the oldest handle information will be overwritten. This can cause a handle to no longer be valid.

Object identification size. The amount of data provided for the LOBJ0200 format. This field must be set to 68.

Filter Format

The format of the lock filter used on the returned lock information.

LKFL0100 Format

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Filter size
4	4	BINARY(4)	Filter lock state
8	8	BINARY(4)	Filter lock scope
12	C	BINARY(4)	Filter lock status
16	10	CHAR(1)	Filter lock holder type
17	11	CHAR(1)	Filter member lock type

Field Descriptions

Filter lock holder type. Filters information that is returned so it contains only information about a type of lock holder.

0   Do not filter on holder type.

1   Return only locks with a holder type of job or thread.

2   Return only locks with a holder type of lock space.

Default   Do not filter on holder type.

Filter lock scope. Filters information that is returned so it contains only information about locks that have a certain lock scope.

0   Do not filter on lock scope.

1   Return only the job scope locks.

2   Return only the thread scope locks.

3   Return only the lock space scope locks.

Default   Do not filter on lock scope.

Filter lock state. Filters information that is returned so it contains only information about locks that have a certain lock state.

0   Do not filter on lock state.

1   Return only the shared locks.

2   Return only the exclusive locks.

Default   Do not filter on lock state.

Filter lock status. Filters information that is returned so it contains only information about locks that have a certain lock status.

0   Do not filter on lock status.

1   Return only locks with a status of held.

2   Return only locks with a status of waiting.

3   Return only locks with a status of requested.

Default   Do not filter on lock status.

Filter member lock type. Filters information that is returned so it contains only information about locks that have a certain member lock type.

0   Do not filter on member lock type.

1   Return only locks with member lock type of member.

2   Return only locks with member lock type of data.

3   Return only locks with member lock type of access path.

Default   Do not filter on member lock type.

Filter size. The size of the filter information passed. Valid values are:

4   No filtering will be performed.

18 All filters will be required.

Lock Information Format

The formats for the lock information.

LCKI0100 Format

Header Section

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Type of entity
12	C	CHAR(30)	Extended object name returned
42	2A	CHAR(10)	Object library name returned
52	34	CHAR(10)	Object ASP name returned
62	3E	CHAR(10)	Object library ASP name returned
72	48	BINARY(4)	Object ASP number returned
76	4C	BINARY(4)	Object library ASP number returned
80	50	CHAR(10)	Object type returned
90	5A	CHAR(10)	Extended object attribute returned
100	64	BINARY(4)	Number of lock information entries available
104	68	BINARY(4)	Offset to list of lock information entries
108	6C	BINARY(4)	Number of lock information entries returned
112	70	BINARY(4)	Length of lock information entry

Lock Information Entry Format

Offset		Type	Field
Dec	Hex		
These fields repeat, in the order listed.		CHAR(10)	Lock state
		CHAR(2)	Reserved
		BINARY(4)	Lock status
		CHAR(1)	Lock scope
		CHAR(3)	Reserved
		CHAR(20)	Lock space identifier
		CHAR(64)	Lock request handle
		BINARY(4)	Lock count
		CHAR(10)	Member name returned
		CHAR(1)	Member lock type
		CHAR(1)	Reserved
		BINARY(4)	Relative record number
		BINARY(4)	Displacement from lock information to holder identification

	BINARY(4)	Displacement from lock information to key information
	BINARY(4)	Number of keys returned
	BINARY(4)	Holder type
	CHAR(*)	Holder identification

Key information format

Offset		Type	Field
Dec	**Hex**		
These fields repeat, in the order listed, for each key selected.		BINARY(4)	Length of field information returned
		BINARY(4)	Key field
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data
		CHAR(*)	Data
		CHAR(*)	Reserved

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. Only complete lock information entries are returned.

Data. The data returned for the key field.

Displacement from lock information to holder identification. The byte displacement from the start of the current lock information entry to the holder identification.

Displacement from lock information to key information. The byte displacement from the start of the current lock information entry to the key information requested.

Extended object attribute returned. The extended attribute of the object for which the list of locks is returned, such as a program or a file type. Extended attributes further describe the object. For example, an object type of *FILE may have an extended object attribute of PHY (physical file), LGL (logical file), DSP (display file), SAV (save file), and so forth.

Extended object name returned. The name of the object for which the lock information entries are placed in the list. This field will be blank if the name is for an internal system object or an internal system object space location, and the user does not have *JOBCTL special authority.

Holder identification. This field contains information about the lock holder. This can be either a qualified job name as described in Job Format specifier or a lock space identifier as described in Lock Space Format specifier.

Holder type. This value is set to indicate what type of holder is holding or is waiting to hold a lock on the specified object.

  0   The lock holder is a job or thread. See the Job Format for more information.

  1   The lock holder is a lock space. See the Lock Space Format for more information.

Key field. The field returned. See Valid Keys for the list of valid keys.

Length of data. The length of the data returned for the field.

Length of field information returned. The total length of information returned for this field. This value is used to increment to the next field in the list.

Length of lock information entry. The length of a lock information entry. This value is used to increment to the next entry.

Lock count. The number of identical locks.

Lock request handle. This value is used to identify a handle to lock request process information. Using the Retrieve Lock Request Information (QWCRLRQI) API and passing in this handle you can retrieve additional information about the program that requested this lock. See QWCRLRQI API for additional information

Lock scope. The scope of the lock. The possible values are:

  0   Job scope

  1   Thread scope

  2   Lock space scope

Lock space identifier. This field contains a value only when the lock scope value is lock space scope and the lock is being waited on by a thread. This field will then contain the lock space ID value for which the lock is being waited on.

Lock state. The lock condition for the lock request. The possible values are:

  *SHRRD   Lock shared for read.

  *SHRUPD   Lock shared for update.

  *SHRNUP   Lock shared no update.

  *EXCLRD   Lock exclusive allow read.

  *EXCL   Lock exclusive no read.

  *RECRD   Lock is a record shared read lock

  *RECUP   Lock is a record exclusive update lock.

  *RECINT   Lock is a record shared internal lock

Lock status. The status of the lock. The lock may be a single request or part of a multiple lock request for which some other object specified in the request has been identified as unavailable. The possible values are:

1   The lock is currently held by the job or thread.

2   The job or thread is waiting for the lock (synchronous).

3   The job or thread has a lock request outstanding for the object (asynchronous).

Member lock type. If the lock is on a member then this field indicates the type of member lock, otherwise it will be blank. The possible values are:

Blank   The object type is not a member

1   Lock on the member control block

2   Lock on the actual data within the member

3   Lock on the access path used to access a member's data

Member name returned. The name of the member that lock information is being returned for. Blanks will be returned if the object is not a member of a database file.

Number of keys returned. The number of keys that were returned.

Number of lock information entries available. The number of lock information entries that were found.

Number of lock information entries returned. The number of lock information entries that are returned.

Object ASP name returned. The name of the ASP in which the object is located. The following special values may also be returned:

*SYSBAS   The object is located in the system ASP or a basic user ASP.

*N   The name of the ASP cannot be determined.

Object ASP number returned. The numeric identifier of the ASP containing the locked object. The following values may be returned:

1   The object is located in the system ASP.

2-32   The object is located in a basic user ASP.

33-255   The object is located in an independent ASP.

-1   The ASP number cannot be determined.

Object library ASP name returned. The name of the ASP in which the object's library is located. The following special values may also be returned:

*SYSBAS   The library is located in the system ASP or a basic user ASP.

*N   The name of the ASP cannot be determined.

Object library ASP number returned. The numeric identifier of the ASP device containing the object's library. The following special values may also be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255	The library is located in an independent ASP.
-1	The ASP number cannot be determined.

Object library name returned. The name of the library that contains the object whose lock information entries are placed in the list.

Object type returned. The type of object for which locks are retrieved. For a list of all the available external OS/400 object types, see the [Control Language (CL)](#) topic. For a list of all internal system object types, see the OS/400 Diagnostic Aids.

Offset to list of lock information entries. The offset in bytes from the beginning of the receiver variable to the first entry.

Relative record number. The relative record number for which record lock information is being returned. If this is not a record lock then this value will be zero.

Reserved. An unused field.

Type of data. The type of data returned.

C	The data is returned in character format.
B	The data is returned in binary format.

Type of entity. This is a value that will identify the type of entity for which the lock information is returned. The following values may be returned:

1	OS/400 external object
2	Member object
3	OS/400 external object space location
4	Internal system object
5	Internal system object space location

Valid Keys

The following table contains a list of valid keys.

See [Field Descriptions](#) in the Work Management API Attributes Descriptions for the descriptions of the valid key fields. All fields are scoped to the job unless specifically noted. See [Considerations for Attribute Scope and Thread Safety](#) in the Work Management API Attributes Descriptions for complete information.

Note: If the holder type is a lock space holder then character fields will contain blanks and the binary fields will be zero.

Key	Type	Description
0101	CHAR(4)	Active job status
0103	CHAR(4)	Active job status for jobs ending

0502	CHAR(1)	End status
0601	CHAR(10)	Function name
0602	CHAR(1)	Function type
0902	CHAR(16)	Internal job identifier
1014	BINARY(4)	Job end reason
1307	CHAR(1)	Message reply
2010	CHAR(4)	Thread status

Holder ID

This field contains an identifier for the job or lock space that is holding or waiting to hold a lock on the specified object or space location.

Job Format

For a job, the returned data in this field will be in this format.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Holder identification size
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(8)	Thread identifier
42	2A	CHAR(2)	Reserved
44	2C	UNSIGNED BINARY(4)	Thread handle

Field Descriptions

Holder identification size. The size of the holder identification.

Job name. The simple job name of the job that issued the lock request. If the job name is MACHINE and the job user name entry is blank, the lock is held by an internal machine process.

Job number. The system assigned job number of the job that issued the lock request. This value will be blank if the job name is MACHINE and the user name entry is blank.

Thread handle. A value which addresses a particular thread within a job holding a thread scope lock or the thread waiting for a lock, otherwise this is zero. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread.

Thread identifier. A value which uniquely identifies a thread within a job holding a thread scope lock or

the thread waiting for a lock, otherwise this is zero.

Reserved. This field will be blank.

User name. The user name under which the job that issued the lock request is started. If the job name is MACHINE and the lock is held by an internal machine process then this field will be blank.

Lock Space Format

For a lock space the returned data in this field will be in this format.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Holder identification size
4	4	CHAR(20)	Holder lock space identifier

Field Descriptions

Holder identification size. The size of the holder identification.

Holder lock space identifier. The identifier of the lock space that holds a lock.

Reserved. A blank field.

Error Messages

Message ID	Error Message Text
CPF0951 E	QSYS only valid library for object type &2.
CPF18C2 E	Value for handle has expired.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C31 E	Object type &1 is not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred while running API &1.
CPF9801 E	Object &1 type &3 not found in library &2.

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

«

API introduced: V5R2

»Retrieve Lock Request Information (QWCRLRQI) API

```
Required Parameter Group:

    1    Receiver variable                    Output        Char(*)
    2    Length of receiver variable          Input         Binary(4)
    3    Format of lock request information   Input         Char(8)
    4    Lock request handle                  Input         Char(64)
    5    Error code                           I/O           Char(*)


Default Public Authority: *USE

Threadsafe: Yes
```

The Retrieve Lock Request Information (QWCRLRQI) API takes as input a lock request handle that was returned in other APIs and returns information about the program that requested the lock. This API must be called from the same thread that called the API that returned the lock request handle.

The handle is a temporary value. A storage limitation will allow a thread to create up to 1 million valid handles. Once the storage limit has been reached the oldest handle information will be overwritten. This can cause a handle to no longer be valid.

Authorities and Locks

None.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

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

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of the receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of the receiver variable specified is larger then the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of lock request information

> INPUT; CHAR(8)

The format of the information returned in the receiver variable. The possible format names are:

LRQI0100   Lock request information. See [Format LRQI0100](#) for details.

Lock request handle

INPUT; CHAR(64)

This contains the handle that will be used to locate the lock request information. This value is created by a previous call to another lock API. The previous lock API call must have been made by the current thread. This value is a temporary value and may become invalid.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

Format LRQI0100

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Offset to statement identifiers
12	C	BINARY(4)	Number of statement identifiers available
16	10	BINARY(4)	Number of statement identifiers returned
20	14	BINARY(4)	Offset to procedure name
24	18	BINARY(4)	Length of procedure name
28	1C	CHAR(10)	Program name
38	26	CHAR(10)	Program library name
48	30	CHAR(10)	Program ASP name
58	3A	CHAR(10)	Program library ASP name
68	44	BINARY(4)	Program ASP number
72	48	BINARY(4)	Program library ASP number
76	4C	BINARY(4)	MI instruction number
80	50	CHAR(10)	Module name
90	5A	CHAR(10)	Module library name
*	*	ARRAY OF CHAR(10)	Statement identifiers
*	*	CHAR(*)	Procedure name

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. Only complete fields are returned.

Length of procedure name. The length of the procedure name in bytes. This field is zero if the program is not an ILE program.

MI instruction number. The current machine instruction number in the program. This field is not meaningful for integrated language environment (ILE) procedures. A zero is returned for ILE procedures.

Module library name. The name of the library in which the module is located. The following special values may be returned:

N   The module library name is unavailable. Either the program has been destroyed or the library containing the program is locked.

blanks   The program is not an ILE program.

Module name. The module containing the integrated language environment (ILE) procedure. The following special values may be returned:

N   The module name is unavailable. Either the program has been destroyed or the library containing the program is locked.

blanks   The program is not an ILE program.

Number of statement identifiers available. The number of statement identifiers available. This field is zero if the program or procedure was not created with debugging tables.

Number of statement identifiers returned. The actual number of statement identifiers returned.

Offset to procedure name. The offset in bytes from the beginning of the receiver variable to the procedure name. This value will be zero if the program is not an ILE program.

Offset to statement identifiers. The offset in bytes from the beginning of the receiver variable to the statement identifiers. This value will be zero if no statement identifiers are returned.

Procedure name. The name of the procedure.

Program ASP name. The name of the auxiliary storage pool (ASP) device in which the program is located. The following special values may also be returned:

SYSBAS   The program is located in the system ASP or a basic user ASP.

N   The name of the ASP cannot be determined.

Program ASP number. The numeric identifier of the ASP containing the program. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP device cannot be determined.

Program library ASP name. The name of the ASP in which the program library is located. The following special values may also be returned:

*SYSBAS   The program library is located in the system ASP or a basic user ASP.

*N   The name of the ASP cannot be determined.

Program library ASP number. The numeric identifier of the ASP containing the program library. The following values may be returned:

1   The library is located in the system ASP or in a basic user ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP device cannot be determined.

Program library name. The name of the library in which the program is located. The following special value may be returned:

*N   The program library name is unavailable. The library containing the program has been destroyed or is locked.

Program name. The name of the program. This can be any type of program object, including objects of type *PGM and *SRVPGM. The following special value may be returned:

*N   The program is unavailable. Either the program has been destroyed or the library containing the program is locked.

Statement identifiers. The high-level language statement identifier. If this field contains the character representation of a number, the number is right-adjusted in the field and padded on the left with zeros (for example, '0000000246'). If the lock is for an integrated language environment (ILE) procedure, more than one statement identifier may exist because of the compilers used for ILE languages.

Error Messages

Message ID	Error Message Text
CPF18C2 E	Object lock handle or lock request handle not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.

CPF3CF2 E   Error occured during running of &1 API.

«

API introduced: V5R2

»Retrieve Lock Space Attributes (QTRXRLSA) API

```
Required Parameter Group:


    1    Receiver variable                Output      Char(*)
    2    Length of receiver variable      Input       Binary(4)
    3    Format of receiver information    Input       Char(8)
    4    Lock space identifier             Input       Char(20)
    5    Error code                        I/O         Char(*)


Default Public Authority: *USE

Threadsafe: Yes
```

The Retrieve Lock Space Attributes (QTRXRLSA) API returns information for the specified lock space. A lock space is an internal object that is used by other objects to hold object and record locks.

Authorities and Locks

Job authority

> The caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is to receive the lock space attributes. The size of this variable is specified in the length of receiver variable parameter.

> See [Format of receiver information](#) for details on the format of the receiver information.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of receiver information

> INPUT; CHAR(8)

The format of the information returned in the receiver variable. The format name is:

RLSA0100   Lock space attributes. See RLSA0100 Format for details.

Lock space identifier

INPUT; CHAR(20)

The identifier of the lock space for which attributes are to be returned.

Error code

I/O; CHAR(*)

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

Format of receiver information

The format of the information returned in the receiver variable.

RLSA0100 Format

The following information is returned for the RLSA0100 format. For detailed descriptions of the fields in the table, see Field Descriptions for RLSA0100 Format.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Lock space type
12	C	BINARY(4)	Lock space state
16	10	BINARY(8)	Lock wait time
24	18	BINARY(8)	Active state timer
32	20	BINARY(4)	Number of threads with lock space attached
36	24	BINARY(4)	Maximum number of threads with lock space attached
40	28	CHAR(8)	Reserved
48	30	CHAR(30)	Lock space name
78	4E	CHAR(10)	Lock space library name
88	58	CHAR(10)	Lock space ASP name
98	62	CHAR(10)	Lock space library ASP name
108	6C	BINARY(4)	Lock space ASP number
112	70	BINARY(4)	Lock space library ASP number

Field Descriptions for RLSA0100 Format

Active state timer. The maximum amount of time, in seconds, that a lock space can remain in the active state before it is switched to the disabled state. The following special value may be returned:

0   The active state timer is disabled for the lock space.

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.

Lock space ASP name. The name of the auxiliary storage pool (ASP) that contains the lock space. The following special values also may be returned:

*SYSBAS   The lock space is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Lock space ASP number. The numeric identifier of the ASP containing the lock space. The following values may be returned:

1   The lock space is located in the system ASP.

2-32   The lock space is located in a basic user ASP.

33-255   The lock space is located in an independent ASP.

-1   The ASP number cannot be determined.

Lock space library ASP name. The name of the auxiliary storage pool (ASP) that contains the library. The following special values also may be returned:

*SYSBAS   The library is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Lock space library ASP number. The numeric identifier of the ASP containing the library. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP number cannot be determined.

Lock space library name. The name of the library that contains the lock space.

Lock space name. The name of the lock space.

Lock space state. The current state of the lock space. The lock space state is used to determine whether object and record locks can be obtained or held by the lock space. The possible values are:

0   Inactive. The lock space cannot hold object or record locks.

1   Active. The lock space can hold object or record locks.

2   Disabled. The lock space can hold object or record locks, but new object or record locks cannot be
        obtained.

Lock space type. The type of object that is using the lock space to hold object and record locks. The
possible values are:

1   Lock space used for UDB-managed commitment definitions.

2   Lock space used for externally managed commitment definitions with job scoped locks.

3   Lock space used for externally managed commitment definitions with lock space scoped locks.

-1   The lock space type cannot be determined.

Lock wait time. The maximum anount of time, in seconds, that a thread is allowed to wait for an object or
record lock that is to be held by the lock space. The following special values may be returned:

0   The lock wait time for the lock space is ignored. The lock wait time specified on the lock request is
        used.

-1   The lock wait is indefinite.

-2   The lock request will return immediately if the lock cannot be obtained.

Maximum number of threads with lock space attached. The maximum number of threads to which the
lock space can be attached concurrently. The following special value may be returned:

-1   There is no limit to the number of threads to which the lock space can be attached.

Number of threads with lock space attached. The number of threads to which the lock space is currently
attached.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.

CPF3CF2 E   Error(s) occurred during running of &1 API.

CPFBDD1 E   Lock space &1 not found.

CPFBDD2 E   No authority to lock space &1.

«

API introduced: V5R2

»Retrieve Lock Space Locks (QTRXRLSL) API

<table>
<tr><td colspan="4">Required Parameter Group:</td></tr>
<tr><td>1</td><td>Receiver variable</td><td>Output</td><td>Char(*)</td></tr>
<tr><td>2</td><td>Length of receiver variable</td><td>Input</td><td>Binary(4)</td></tr>
<tr><td>3</td><td>Format of receiver information</td><td>Input</td><td>Char(8)</td></tr>
<tr><td>4</td><td>Lock space identifier</td><td>Input</td><td>Char(20)</td></tr>
<tr><td>5</td><td>Lock filters</td><td>Input</td><td>Char(*)</td></tr>
<tr><td>6</td><td>Format of lock filters</td><td>Input</td><td>Char(8)</td></tr>
<tr><td>7</td><td>Error code</td><td>I/O</td><td>Char(*)</td></tr>
</table>

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Lock Space Locks (QTRXRLSL) API generates a list of objects that have been locked or that have lower level locks held by the specified lock space. Locks that are being waited for on behalf of a lock space are not returned. Use the Retrieve Job Locks (QWCRJBLK) or Retrieve Lock Information (QWCRLCKI) API to retrieve locks that are being waited for by a thread on behalf of a lock space.

Authorities and Locks

Job Authority

The caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold. For example, this may mean that the number of locked object entries available in the receiver variable do not match the value in the number of locked object entries returned.

See Format of Receiver Information for details on the format of the receiver information.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable provided. The length of the receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of the receiver variable specified is larger then the allocated size of the receiver variable specified in the

user program, the results are not predictable. The minimum length is 8 bytes.

Format of receiver information

      INPUT; CHAR(8)

      The format of the information returned in the receiver variable. The format name is:

         RLSL0100   The object and member lock format. See [RLSL0100 Format](#) for details on the list of objects and members that this lock space has locked.

Lock space identifier

      INPUT; CHAR(20)

      The identifier of the lock space for which record locks are to be returned.

Lock filters

      INPUT;CHAR(*)

      Filters used for the lock information that is returned. See [Format of lock filters](#) for further information.

Format of lock filters

      INPUT; CHAR(8)

      The format of the lock filters used on the returned data. The format name is:

         RLSF0100   Lock filter format. See [RLSF0100 Format](#) for details on the filters contained in this format.

Error code

      I/O; CHAR(*)

      The structure in which to return error information. For the format of the structure, see [Error code parameter](#).

Format of receiver information

The format of the information returned in the receiver variable.

RLSL0100 Format

This format is used to return objects and members that are locked. For detailed descriptions of the fields in the table, see [Field Descriptions](#) for RLSL0100 Format.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available

8	8	BINARY(4)	Number of locked object entries available
16	10	BINARY(4)	Offset to list of locked objects
12	C	BINARY(4)	Number of locked object entries returned
20	14	BINARY(4)	Length of locked object entry

Each locked object returned will have the following structure.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Type of entity
4	4	CHAR(30)	Extended object name
34	22	CHAR(10)	Object library name
44	2C	CHAR(10)	Object ASP name
54	36	CHAR(10)	Object library ASP name
64	40	BINARY(4)	Object ASP number
68	44	BINARY(4)	Object library ASP number
72	48	CHAR(10)	Object type
82	52	CHAR(10)	Extended object attributes
92	5C	CHAR(10)	Member name
102	66	CHAR(1)	Member lock type
103	67	CHAR(3)	Reserved
106	6A	CHAR(10)	Lock state
116	74	BINARY(4)	Lock status
120	78	BINARY(4)	Member locks
124	7C	BINARY(4)	Lock count
128	80	CHAR(64)	Object lock handle
192	C0	CHAR(64)	Lock request handle

Field Descriptions for RLSL0100 Format

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. Only complete entries are returned.

Extended object attributes. The extended attributes of an object. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG (RPG program) or CLP (CL program). This field will be blank if there is no extended attribute associated with the object type.

Extended object name. The name of the object that is locked. If the lock is on a database member, the object name is the name of the file that owns the member to which the lock applies. If the member lock type is member or access path, the file that owns the member may be either a physical file or a logical file. If the member lock type is data, the file that owns the member is a physical file. The following special value also

may be returned:

N   The name of the object cannot be determined.

Length of locked object entry. The length of each locked object entry.

Lock count. The number of identical locks on this entity.

Lock request handle. A handle to lock request information. Using the Retrieve Lock Request Information (QWCRLRQI) API and passing in this handle, you can retrieve additional information about the program that requested this lock. A value of hexadecimal zero is returned when additional information cannot be retrieved. This value is a temporary value that can expire. See the Retrieve Lock Request Information (QWCRLRQI) API for additional information.

Lock state. The lock condition for the lock request. Lower level locks are returned and can occur when a member of a file is locked, but the file itself is not locked. Possible other values are:

Blank   The object is not locked, but there are locks on lower level objects.

*SHRRD   Lock shared for read.

*SHRUPD   Lock shared for update.

*SHRNUP   Lock shared, no update.

*EXCLRD   Lock exclusive, read allowed.

*EXCL   Lock exclusive, no read allowed.

Lock status. The status of the lock request. Possible values are:

0   The object is not locked, but there are locks on lower level objects.

1   The lock on this object currently is held by the lock space.

Member locks. The number of member locks for a database file. If the object is not a database file, 0 is returned.

Member lock type. If the lock is on a member, this field indicates the type of member lock; otherwise, it is blank. Possible values are:

Blank   The object type is not a member.

0   The lock is a member lock.

1   The lock is a data lock.

2   The lock is an access path lock.

Member name. The name of the member that has a lock held or waiting on it. If the type of entity is not a member object, this field is blank. The following special value also may be returned:

N   The name of the object cannot be determined.

Number of locked object entries available. The number of locked object entries that are held by this lock space.

Number of locked object entries returned. The number of locked object entries that are returned.

Object ASP name. The name of the auxiliary sptorage pool (ASP) that contains the object that is locked. The following special values also may be returned:

SYSBAS   The object is located in the system ASP or a basic user ASP.

N   The name of the ASP device cannot be determined.

Object ASP number. The numeric identifier of the ASP containing the locked object. The following values may be returned:

1   The object is located in the system ASP.

2-32   The object is located in a basic user ASP.

33-255 The object is located in an independent ASP.

-1   The ASP number cannot be determined.

Object library ASP name. The name of the ASP containing the library of the locked object. The following specials value also may be returned:

SYSBAS   The object's library is located in the system ASP or a basic user ASP.

N   The name of the ASP device cannot be determined.

Object library ASP number. The numeric identifier of the ASP containing the library of the locked object. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255 The library is located in an independent ASP.

-1   The ASP number cannot be determined.

Object library name. The name of the library containing the locked object. This field will be blank if the type of entity is an internal system object or is an internal system object space location. The following special value also may be returned:

N The name of the library cannot be determined.

Object lock handle. An identifier that can be input to Retrieve Lock Information (QWCRLCKI) API to find additional information about other holders of locks on this object. A value of hexadecimal zero is returned when additional information cannot be retrieved. The object lock handle is a temporary value that can expire. See the QWCRLCKI API for additional information.

Object type. The object type. For a list of all the available external OS/400 object types, see the [Control Language (CL)](#) topic.

Offset to list of locked objects. The offset in bytes from the beginning of the receiver variable to the first entry.

Reserved. An unused field.

Type of entity. A value that identifies the type of entity for which the lock information is returned. The following values may be returned:

1 OS/400 external object

2 Member object

3 Internal system object

Format of lock filters

The format of the lock filters used on the returned lock information.

RLSF0100 Format

The following information is to be specified for the RLSF0100 format. For detailed descriptions of the fields in the table, see Field Descriptions for RLSF0100 Format.

Offset		Type	Field
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Filter size
4	4	BINARY(4)	Filter lock state
8	8	CHAR(1)	Include OS/400 external objects flag
9	9	CHAR(1)	Include member objects flag
10	A	CHAR(1)	Include internal system objects flag
11	B	CHAR(1)	Include lock space objects flag
12	C	CHAR(1)	Include unknown entities flag
13	D	CHAR(1)	Reserved
14	E	CHAR(10)	Filter object name
24	18	CHAR(10)	Filter object library name
34	22	CHAR(10)	Filter object library ASP name

Field Descriptions for RLSF0100 Format

Filter lock state. Filters information that is returned so that it only contains information about locks that have a certain lock state. The default is do not filter on lock state value.

0 Do not filter on lock state value.

1 Return only the shared locks.

2 Return only the exclusive locks.

Filter object library ASP name. The name of the library's auxiliary storage pool (ASP) on which to filter. A special value of *SYSBAS can be specified. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter object library name. This is the library name on which to filter. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter object name. Only locks on the specified object are returned. In the case of database files, members of the file also may be returned. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter size. The size of the filter information passed. Valid values are:

4   No filtering will be performed. The default values will be used for each filter.

44 All filters will be required.

Include internal system objects flag. A value of 1 in this field allows internal system object locks to be returned. A value of 0 causes these values to be excluded from the return data. The default is to exclude internal system objects.

Include lock space objects flag. A value of 1 in this field allows lock space objects locks to be returned. A value of 0 causes these values to be excluded from the return data. The default is to include lock space objects.

Include member objects flag. A value of 1 in this field allows member objects locks to be returned. A value of 0 causes these values to be excluded from the return data. The default is to include member objects.

Include OS/400 external objects flag. A value of 1 in this field allows OS/400 external object locks to be returned. A value of 0 causes these objects to be excluded from the return data. The default is to include OS/400 external objects.

Include unknown types flag. A value of 1 in this field allows locks that are of an unknown entity type to be returned. A value of 0 causes these values to be excluded from the return data. The default is to exclude unknown types.

Reserved. An unused field.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.

CPF3CF2 E   Error(s) occurred during running of &1 API.

CPFBDD1 E   Lock space &1 not found.

CPFBDD2 E   No authority to lock space &1.

《

API introduced: V5R2

»Retrieve Lock Space Record Locks (QTRXRLRL) API

```
Required Parameter Group:

    1    Receiver variable              Output      Char(*)
    2    Length of receiver variable    Input       Binary(4)
    3    Format of receiver information  Input       Char(8)
    4    Lock space identifier          Input       Char(20)
    5    Lock filters                   Input       Char(*)
    6    Format of lock filters         Input       Char(8)
    7    Error code                     I/O         Char(*)


Default Public Authority: *USE

Threadsafe: Yes
```

The Retrieve Lock Space Record Locks (QTRXRLRL) API lets you generate a list of record locks held by the specified lock space. Record locks that are being waited for on behalf of a lock space are not returned. Use the Retrieve Job Record Locks (QDBRJBRL) or Retrieve Lock Information (QWCRLCKI) API to retrieve record locks that are being waited for by a thread on behalf of a lock space.

Lock information is returned for local physical files only. The Retrieve Lock Space Record Locks API places the list in the specified receiver variable.

Authorities and Locks

Job authority

> The caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is to receive the list of record locks. The size of this variable is specified in the length of receiver variable parameter.

> See Format of receiver information for details on the format of the receiver information.

Length of receiver variable

> INPUT; BINARY(4)

> The number of bytes that are provided in the Receiver variable parameter. At least 16 bytes must be

provided. If the size of the receiver variable provided is less than the length of the list that is available, the list will be truncated; this can be determined by examining the first two fields in the receiver variable, the number of record locks returned, and the number of record locks available. If the receiver variable length specified is greater than the actual receiver variable, the results are unpredictable.

Format of receiver information

> INPUT; CHAR(8)

> The format of the information returned in the receiver variable. The format name is:

> > *RLRL0100*   Record lock list. See [RLRL0100 Format](#) for details.

Lock space identifier

> INPUT; CHAR(20)

> The identifier of the lock space for which record locks are to be returned.

Lock filters

> INPUT;CHAR(*)

> Filters used for the lock information that is returned. See [Format of lock filters](#) for further information.

Format of lock filters

> INPUT; CHAR(8)

> The format of the lock filters used on the returned data. The possible format name is:

> > *RLRF0100*   Lock filter format. See [RLRF0100 Format](#) for details.

Error code

> I/O; CHAR(*)

> The structure in which to return error information. For the format of the structure, see [Error code parameter](#).

Format of receiver information

The format of the information returned in the receiver variable.

RLRL0100 Format

The following information is returned for the RLRL0100 format. For detailed descriptions of the fields in the table, see [Field Descriptions](#) for RLRL0100 Format.

Offset			
Dec	Hex	Type	Field

0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of record locks available
12	C	BINARY(4)	Number of record locks returned
16	10	BINARY(4)	Offset to list of record locks
20	14	BINARY(4)	Size of information for each record lock returned

Each record lock returned will have the following structure.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Database file name
10	A	CHAR(10)	Database file library name
20	14	CHAR(10)	Database member name
30	1E	CHAR(1)	Reserved
31	1F	CHAR(1)	Lock state
32	20	UNSIGNED BINARY(4)	Relative record number
36	24	CHAR(10)	Database file ASP name
46	2E	CHAR(10)	Database file library ASP name
56	38	BINARY(4)	Database file ASP number
60	3C	BINARY(4)	Database file library ASP number

Field Descriptions for RLRL0100 Format

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. Only complete entries are returned.

Database file library ASP name. The name of the auxiliary storage pool (ASP) that contains the library. The following special values also may be returned:

SYSBAS   The library is located in the system ASP or a basic user ASP.

N   The name of the ASP device cannot be determined.

Database file library ASP number. The numeric identifier of the ASP containing the library. The following values may be returned:

1   The library is located in the system ASP.

2-32   The library is located in a basic user ASP.

33-255   The library is located in an independent ASP.

-1   The ASP number cannot be determined.

Database file library name. The name of the library that contains the file.

Database file name. The name of the file.

Database member name. The name of the member.

Database file ASP name. The name of the auxiliary storage pool (ASP) that contains the file. The following special values also may be returned:

*SYSBAS   The file is located in the system ASP or a basic user ASP.

*N   The name of the ASP device cannot be determined.

Database file ASP number. The numeric identifier of the ASP containing the file. The following values may be returned:

1   The file is located in the system ASP.

2-32   The file is located in a basic user ASP.

33-255   The file is located in an independent ASP.

-1   The ASP number cannot be determined.

Lock state. The state of the lock. The possible values are:

0   Shared Read.

1   Exclusive Update.

2   Shared Internal.

Number of record locks available. The number of record lock structures that are available to be returned. If this field is the same as the number of record locks returned field, all the record lock information has been returned.

Number of record locks returned. The number of record lock structures that were returned to the caller of the API. If enough space is provided in the receiver variable, all record locks are returned. If there is more record lock information than can fit in the space provided, the number of record locks returned is less than the number of record locks available.

Offset to list of record locks. The byte offset from the beginning of the receiver variable to the first record lock information structure.

Relative record number. The relative record number for which record lock information is being returned.

Reserved. An unused field.

Size of information for each lock returned. The number of bytes of each of the returned lock information structures. In future releases, the amount of information returned for each lock may be expanded, so this value should be used to move from one lock structure to another.

Format of lock filters

The format of the lock filters used on the returned lock information.

RLRF0100 Format

The following information is to be specified for the RLRF0100 format. For detailed descriptions of the fields in the table, see [Field Descriptions](#) for RLFL100 Format.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Filter size
4	4	BINARY(4)	Filter lock state
8	8	CHAR(10)	Filter file name
18	12	CHAR(10)	Filter file member name
28	1C	CHAR(10)	Filter file library name
38	26	CHAR(10)	Filter file library ASP name

Field Descriptions for RLRF0100 Format

Filter lock state. Filters information that is returned so that it contains only information about locks that have a certain lock state. The default is do not filter on lock state value.

0 Do not filter on lock state value

1 Return only the shared locks

2 Return only the exclusive locks

Filter file library ASP name. The name of the library's auxiliary storage pool (ASP) on which to filter. A special value of *SYSBAS can be specified. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter file library name. The library name on which to filter. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter file member name. The member name on which to filter. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter file name. The file name on which to filter. A blank field will cause no filtering to be done on this field. The default is to not filter on this field.

Filter size. The size of the filter information passed. Valid values are:

4 No filtering will be performed. The default values will be used for each filter.

48 All filters will be required.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPFBDD1 E	Lock space &1 not found.
CPFBDD2 E	No authority to lock space &1.

《

API introduced: V5R2

Retrieve Network Attributes (QWCRNETA) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Number of network attributes to retrieve	Input	Binary(4)
4	Network attribute names	Input	Array(*) of Char(10)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Network Attributes (QWCRNETA) API lets you retrieve network attributes.

Authorities and Locks

None.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is to receive the information requested. For the format, see Format of Data Returned.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable described in Format of Data Returned. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 28 bytes.

Number of network attributes to retrieve

> INPUT; BINARY(4)

> The total number of network attributes to be retrieved.

Network attribute names

> INPUT: ARRAY(*) of CHAR(10)

> The names of the network attributes to be retrieved. This can be a list of network attribute names where each name is 10 characters.

Error code

I/O; CHAR(*)

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

Format of Data Returned

The receiver variable holds the information returned about each network attribute.

The receiver variable has three logical parts:

1. The first field specifies the number of network attributes returned.
2. The next fields give the offsets to the network attributes returned. There is one offset field for each network attribute returned.
3. Next are the network attribute information tables for the network attributes returned. There is one network attribute information table for each network attribute.

The following table shows the format of the receiver variable. The offset fields are repeated until the offsets for all the network attributes returned are listed; the network attribute information table for each network attribute is repeated in the same way. For a detailed description of each field, see the Field Descriptions.

The format of the receiver variable is:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of network attributes returned
4	4	ARRAY(*) of BINARY(4)	Offset to network attribute information table
*	*	CHAR(*)	Network attribute information table
Note: Each network attribute in the table is represented by the standard network attribute information table described in Network Attribute Information Table.			

To determine the length of the receiver variable, the following calculation should be done. For each network attribute to be returned, get the length of the data returned for the network attribute and add 24. After adding the values for each network attribute, add 4. This calculation takes into account the data alignment that needs to be done; therefore, this value is a worst-case estimate.

Network Attribute Information Table

The following table shows the format of the network attribute information table.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Network attribute
10	A	CHAR(1)	Type of data
11	B	CHAR(1)	Information status
12	C	BINARY(4)	Length of data
16	10	CHAR(*)	Data

Field Descriptions

Data. The data returned for the network attribute.

Information status. Whether the information was available for the network attribute.

blank The information was available.

L The information was not available because the network attribute was locked.

Length of data. The length of the data returned for the network attribute. If this value is 0, the network attribute was not available.

Network attribute. The network attribute to be retrieved. See Valid Network Attributes for the list of valid network attributes.

Number of network attributes returned. The number of network attributes returned to the application.

Offset to network attribute information table. The offset from the beginning of the structure to the start of the network attribute information.

Type of data. The type of data returned.

blank The data was not available.

C The data is returned in character format.

B The data is returned in binary format.

Valid Network Attributes

For a detailed description of each field, see the Network Attribute Field Descriptions.

Network attribute	Type	Description
ALRBCKFP	CHAR(16)	Alert backup focal point
ALRCTLD	CHAR(10)	Alert controller
ALRDFTFP	CHAR(10)	Alert focal point
ALRFTR	CHAR(20)	Alert filter
ALRHLDCNT	BINARY(4)	Alert hold count
ALRLOGSTS	CHAR(7)	Alert logging status
ALRPRIFP	CHAR(10)	Alert primary focal point
ALRRQSFP	CHAR(16)	Alert focal point to request
ALRSTS	CHAR(10)	Alert status
ALWADDCLU	CHAR(10)	Allow add to cluster
ALWANYNET	CHAR(10)	Allow AnyNet support
ALWHPRTWR	CHAR(10)	Allow HPR tower support

ALWVRTAPPN	CHAR(10)	Allow virtual APPN support
VRTAUTODEV	BINARY(4)	Autocreate APPC device limit
DDMACC	CHAR(20)	DDM request access
DFTCNNLST	CHAR(10)	Default ISDN connection list
DFTMODE	CHAR(8)	Default mode
DFTNETTYPE	CHAR(10)	ISDN network type
DTACPR	BINARY(4)	Data compression
DTACPRINM	BINARY(4)	Intermediate data compression
HPRPTHTMR	CHAR(40)	HPR path switch timers
JOBACN	CHAR(10)	Job action
LCLCPNAME	CHAR(8)	Local control point
LCLLOCNAME	CHAR(8)	Local location
LCLNETID	CHAR(8)	Local network ID
MAXINTSSN	BINARY(4)	Maximum sessions
MAXHOP	BINARY(4)	Maximum hop count
MDMCNTRYID	CHAR(2)	Modem country or region ID
MSGQ	CHAR(20)	Message queue
NETSERVER	CHAR(85)	Server network ID
NODETYPE	CHAR(8)	APPN node type
NWSDOMAIN	CHAR(8)	Network server domain
OUTQ	CHAR(20)	Output queue
PNDSYSNAME	CHAR(8)	Pending system name
PCSACC	CHAR(20)	Client Access
RAR	BINARY(4)	Addition resistance
SYSNAME	CHAR(8)	Current system name

Network Attribute Field Descriptions

Addition resistance. The Advanced Peer-to-Peer Networking (APPN) function routes addition resistance for an APPN node type of *NETNODE or *BEXNODE.

Alert backup focal point. Identifies the system that provides alert focal point services if the local system is unavailable and ALRPRIFP is *YES. The backup focal point is only used by systems in the primary sphere of control. The first eight characters are the control point name and the last eight characters are the network ID. *NONE means no backup focal point is defined.

Alert controller. The name of the controller to be used for alerts in a system service control point - physical unit (SSCP-PU) session. This controller is ignored if the system has a focal point (in which case the node is in another system's sphere of control). *NONE means no alert controller is defined.

Alert filter. The name of the filter object that is used by the alert manager when processing alerts. *NONE means no alert filter is being used. The first ten characters are the filter name and the last ten characters are the library name.

Alert focal point. Whether or not the system is an alert default focal point.

NO   The system is not an alert default focal point.

YES   The system is defined to be an alert default focal point and provides focal point services to all nodes in the network that are not being serviced by another focal point.

Alert focal point to request. Specifies the name of the system that is requested to provide focal point services. If a focal point is already defined for the entry point, it is taken away when the new focal point is requested. *NONE means no focal point is requested.

Alert hold count. The maximum number of alerts to be created before the alerts are sent over the System Service Control Point - Physical Unit (SSCP-PU) session. Alerts are held by the system until this number of alerts have been created. If the Alert Controller (ALRCTLD) network attribute is being used to send alerts (SSCP-PU session), alerts will be sent automatically regardless of the ALRHLDCNT network attribute when a switched connection is made for other reasons.

Alert logging status. Specifies which alerts are to be logged:

LOCAL   Only locally created alerts are logged.

RCV   Only Alerts received from other nodes are logged.

ALL   Both locally created alerts and incoming alerts are logged.

NONE   No alerts are logged.

Alert primary focal point. Whether or not the system is an alert primary focal point.

NO   The system is not an alert primary focal point.

YES   The system is defined to be an alert primary focal point and provides focal point services to all nodes in the network that are explicitly defined in the sphere of control.

Alert status. Alert status controls the creation of local alerts. The following is a list of values and their meanings:

ON   Alerts are created by a system for all changeable conditions except *unattended* conditions.

UNATTEND   Alerts are created by the system for all alert conditions including those which have the alert indicator in the message description set to *UNATTEND.

OFF   Alerts are not created by the system.

Allow add to cluster. Whether this system will allow another system to add it as a node in a cluster. The following is a list of values and their meanings:

NONE   No other system can add this system as a node in a cluster.

ANY   Any other system can add this system as a node in a cluster.

RQSAUT   Any other system can add this system as a node in a cluster only after the cluster add request has been authenticated.

Allow AnyNet(R) support. The AnyNet support value is used for the UNIX-type APIs that use the AF_INET address family. The following is a list of values and their meanings:

NO   The system does not allow AF_INET support to be used over an SNA connection.

YES   The system allows AF_INET (AnyNet) support to be used over an SNA connection.

Allow HPR tower support. The HPR transport tower support value is used for APPN session traffic. The following is a list of values and their meanings:

NO   The system does not allow HPR transport tower support to be used with APPN session traffic.

YES   The system allows HPR transport tower support to be used with APPN session traffic.

Allow virtual APPN support. The virtual APPN support value is used to specify whether or not APPC sessions and devices are allowed to use virtual APPN controllers.

NO   The system does not allow APPC sessions or devices to use virtual APPN controllers. If sessions are using HPR transport tower support, they will use virtual APPN controllers regardless of this setting.

YES   The system does allow APPC sessions or devices to use virtual APPN controllers.

Autocreate APPC device limit. The specification for the APPC device limit used for autocreation of devices on virtual APPN controllers.

APPN node type. The Advanced Peer-to-Peer Networking (APPN) node type can have the following values:

ENDNODE   The node does not provide network services to other nodes, but may participate in the APPN network by using the services of an attached network server, or may operate in a peer environment similar to migration end nodes.

NETNODE   The node provides intermediate routing, route selection services, and distributed directory services for local users and to end nodes and migration end nodes that it is serving.

BEXNODE   The node performs as a branch extender node. The node performs as an end node in the backbone APPN network, and performs as a network node server to end nodes within its local domain.

Client Access. The way in which the system processes Client Access requests from other systems.

REJECT   The system rejects every request from Client Access.

OBJAUT   Normal object authorizations are checked for the Client Access request. For example, authorization to retrieve data from a database file for a transfer function request is checked.

REGFAC   The system uses the system's registration facility to determine which exit program (if any) to run. If no exit program is defined for an exit point and this value is specified, *OBJAUT is used.

program library   The name of a user-written validation program that is called each time a Client Access application request comes from a personal computer. The program is passed two parameters: the first describes the PC request (the name of the application and type request); the second is used by the program to indicate to the Client Access application whether or not this PC request should be handled. The first 10 characters are the program name, and the last 10 characters are the library name.

Current system name. The current system name that appears on displays.

Data compression. Whether data compression is used when the system is an SNA end node (the node containing either a primary or secondary LU). This field is used by mode descriptions that specify *NETATR for data compression. The following values are valid:

0　　　　Data compression is not allowed on the session.

-1　　　　Data compression is requested on the session by the local system. However, the request can be refused or changed to a lower compression level by the remote system. Data compression is allowed on the session if requested by the remote system.

-2　　　　Data compression is allowed on the session by the local system if requested by a remote system. The local system does not request that compression be used.

-3　　　　Data compression is required on the session. If the remote system does not change the levels of compression to the local system's exact requested levels, the session is not established. The data compression levels that the local system requires are the specified levels.

line speed　　If either the receiving or sending link has a line speed equal to or less than this specified line speed, this value indicates the need for compression by requesting that the remote systems compress the session data. Otherwise, this value does not indicate to the remote systems that there is a need to compress the data. Possible values range from 1 through 2 147 483 647 bits per second.

If data compression is requested by the remote system, the data compression levels used by the session are the lower of the requested levels and the configured levels (INDTACPR and OUTDTACPR).

DDM request access. The system processes distributed data management (DDM) and Distributed Relational Database Architecture (DRDA) requests from other systems as follows:

*REJECT　　　　The system does not allow DDM or DRDA requests from remote systems. This system can still use DDM or DRDA, however, to access files or SQL tables on remote systems.

*OBJAUT　　　　All requests are allowed and controlled by the object authorization on the system.

program library　　The name of a user-written validation program that is called each time a DDM request is made from a remote system. This program indicates to DDM whether the request should proceed or be ended. System security still applies. The first ten characters are the program name and the last ten characters are the library name.

Default ISDN connection list. The name of the default integrated services digital network (ISDN) connection list object.

Default mode. The default mode name for the system.

HPR path switch timers. The settings for the amount of time, in minutes, to allow for a path switch attempt of a Rapid Transport Protocol (RTP) connection. Four positional values exist to specify the time allowed based on the type of session traffic. Each positional timer will consist of 10 characters within the 40-character field. A description of each element within the field is given below:

Network priority timer (characters 1-10)

　　　　The first element is the network priority timer, which specifies the amount of time in minutes to allow for a path switch attempt of an RTP connection that has network transmission priority or *NONE.

High priority timer (characters 11-20)

> The second element is the high priority timer, which specifies the amount of time in minutes to allow for a path switch attempt of an RTP connection that has high transmission priority or *NONE.

Medium priority timer (characters 21-30)

> The third element is the medium priority timer, which specifies the amount of time in minutes to allow for a path switch attempt of an RTP connection that has medium transmission priority or *NONE.

Low priority timer (characters 31-40)

> The fourth element is the low priority timer, which specifies the amount of time in minutes to allow for a path switch attempt of an RTP connection that has low transmission priority or *NONE.

Intermediate data compression. The level of data compression to request when the iSeries server is an SNA intermediate node. The following are valid values:

0   Does not indicate to the remote systems that there is a need to compress the data.

-1   Indicates the need for compression by requesting that the remote systems compress the session data.

line speed If either the receiving or sending link has a line speed equal to or less than this specified line speed, this value indicates the need for compression by requesting that the remote systems compress the session data. Otherwise, this value does not indicate to the remote systems that there is a need to compress the data. Possible values range from 1 through 2 147 483 647 bits per second.

ISDN network type. The type of integrated services digital network (ISDN) to which the system is attached. Possible values follow:

*ATT   This value is used when attaching to an ISDN that uses an AT&T interface.

*DBP1TR6   This value is used when attaching to an ISDN controlled by Germany's Post Telephone and Telegraph Administration (PPT). (Deutsche Bundespost 1TR6).

*ETSI   This value is used when attaching to the ISDN that uses a European interface (ETSI, also known as EuroISDN standard).

*JAPAN   This value is used when attaching to an ISDN that uses a Japanese interface.

*NISDN   This value is used when attaching to an ISDN that uses a North American interface.

*NORTEL   This value is used when attaching to an ISDN that uses the Northern Telecom interface.

*NORTHAMT1   This value is used when attaching to a channelized North American T1 interface.

Job action. The action that is taken for any input stream received through the SNA distribution services (SNADS) network by the system.

*REJECT   The input stream is rejected by the system. This action allows users to secure their system from input streams received through the network.

FILE   The input stream is filed in the queue of network files received for the user to whom it was sent. That user may then look at the input stream, end it, receive it, or submit it to a job queue.

SEARCH   The table of network job entries is searched to determine the action to be taken for the input stream.

Local control point. The local control point name for the system.

Local location. The default local location name for the system.

Local network ID. The local network ID assigned to the system.

Maximum hop count. The maximum number of times in an SNA distribution services (SNADS) network that a distribution queue entry originating at this node may be received and routed on the path to its final destination. If this number is exceeded, the distribution queue entry is ended. When the distribution queue entry is ended, a feedback status is sent back to the sender if it was requested.

Maximum sessions. The maximum number of advanced program-to-program communications (APPC) intermediate sessions for an Advanced Peer-to-Peer Networking (APPN) node type of *NETNODE or *BEXNODE.

Message queue. The name of the message queue to which messages received through the SNA distribution services (SNADS) network are sent for:

- Users who have no message queue specified in their user profile
- Users whose message queue is not available

The first 10 characters are the message queue name, and the last 10 characters are the library name.

Modem country or region ID. The country or region identifier associated with a modem.

This defines the country or region-specific default characteristics for modems that are internal to OS/400 I/O adapters. This value must be configured correctly to ensure proper operation and, in some countries or regions, meet legal requirements. The adapter will fail the vary on of the line if the modem country or region ID is not set.

Network server domain. The LAN server domain to which all Integrated Netfinity(R) Servers (also known as file server I/O processor and FSIOP) on the system belong.

Output queue. The name of the output queue to which spooled files received through the SNA distribution services (SNADS) network are sent for users whose output queue is not available. The first 10 characters are the output queue name, and the last 10 characters are the library name.

Pending system name. The pending system name (if a change is pending). This will contain blanks if no change is pending.

Server network ID. The network node server of an Advanced Peer-to-Peer Networking (APPN) network (up to a maximum of five) for an APPN node type of *ENDNODE. The list is not used for an APPN node type of *NETNODE or *BEXNODE.

Error Messages

Message ID	Error Message Text
CPF1860 E	Value &1 in list not valid.
CPF1861 E	Length of the receiver variable not valid.
CPF1862 E	Number of values to retrieve not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Retrieve Profile Exit Programs (QWTRTVPX) API

```
Required Parameter Group:


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


Default Public Authority: *USE

Threadsafe: No
```

The Retrieve Profile Exit Programs (QWTRTVPX) API retrieves the profile exit flags, based on the format, that have been designated to be called for the specified user ID. The QWTRTVPX API then places that information into a single variable in the calling program. The amount of information placed in the variable depends on the size of the variable.

Authorities and Locks

User ID Authority

> *READ

Required Parameter Group

Receiver variable

> OUTPUT CHAR(*)

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

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format name

> INPUT; CHAR(8)

The format of the profile exit program information to be returned. You can use this format:

ATTN0100   Preattention information. For details, see [ATTN0100 Format](#).

SREQ0100   Presystem request information. For details, see [SREQ0100 Format](#).

User ID

INPUT; CHAR(10)

The user ID being retrieved. Valid values are as follows:

*CURRENT   The user ID of the job that is currently running.

User ID name   The 10-character name that is entered.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

ATTN0100 Format

The following table describes the information that is returned in the receiver variable for the ATTN0100 format. For detailed descriptions of the fields, see [Field Descriptions](#).

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Entries returned
4	4	BINARY(4)	Entries available
		ARRAY (*) of BINARY(4)	Array of entries

Field Descriptions

Array of entries An array of entries where the first entry corresponds to exit program number one for the exit point QIBM_QWT_PREATTNPGMS and format ATTN0100 in the registration facility. The second entry in the array corresponds to exit program number 2 and so on. The possible returned values are as follows:

0   No, do not call this exit program.

1   Yes, call this exit program.

Entries available The number of possible entries being returned. All available data is returned if enough space is provided.

Entries returned The actual number of entries that is being returned. If the data is truncated because the receiver variable is not large enough to hold all of the data available, this value is less than the entries available.

SREQ0100 Format

The following table describes the information that is returned in the receiver variable for the SREQ0100 format. For detailed descriptions of the fields, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Entries returned
4	4	BINARY(4)	Entries available
		ARRAY (*) of BINARY(4)	Array of entries

Field Descriptions

Array of entries An array of entries where the first entry corresponds to exit program number one for the exit point QIBM_QWT_SYSREQPGMS and format SREQ0100 in the registration facility. The second entry in the array corresponds to exit program number 2 and so on. The possible returned values are as follows:

0   No, do not call this exit program.

1   Yes, call this exit program.

Entries available The number of possible entries being returned. All available data is returned if enough space is provided.

Entries returned The actual number of entries that is being returned. If the data is truncated because the receiver variable is not large enough to hold all of the data available, this value is less than the entries available.

Error Messages

Message ID	Error Message Text
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.

CPF3CF2 E   Error(s) occurred during running of &1 API.

CPF3C21 E   Format name &1 is not valid.

CPF3C24 E   Length of the receiver variable is not valid.

CPF3C90 E   Literal value cannot be changed.

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

API introduced: V3R6

Retrieve Subsystem Information (QWDRSBSD) 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 subsystem name       Input         Char(*)
    5      Error code                     I/O           Char(*)


Optional Parameter:


    6      Number of qualified subsystem  Input         Binary(4)
           libraries


Default Public Authority: *USE

Threadsafe: Yes
```

The Retrieve Subsystem Information (QWDRSBSD) API retrieves information about one or more specific subsystems or all active subsystems.

Authorities and Locks

Subsystem Description Authority

> *USE

Library Authority

> *EXECUTE

Subsystem Description Lock

> *EXCLRD

The subsystem description authority and library authority are not required when *ACTIVE is specified for the qualified subsystem name parameter.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable to receive the subsystem information. This area must be large enough to accommodate the information specified.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. The length must be at least 8 bytes. If the variable is not long enough to hold the subsystem information, the data is truncated.

Format name

INPUT; CHAR(8)

The format of the subsystem information. You can use this format:

SBSI0100   Basic subsystem information. For details, see [SBSI0100 Format](#).

SBSI0200   Multiple subsystems information. For details, see [SBSI0200 Format](#). This format name must be specified if *ACTIVE is specified in the first 10 characters of the qualified subsystem name or if the number of qualified subsystem names parameter is greater than 1.

Qualified subsystem name

INPUT; ARRAY(*) of CHAR(20)

An array of CHAR(20) values giving the names of the subsystems about which to retrieve information and the library in which the subsystem description is located. The number of qualified subsystem names parameter specifies how many elements are in this array.

The first 10 characters contain the subsystem name. You can use the following special value for the subsystem name:

*ACTIVE   Return information about all active subsystems. If *ACTIVE is specified for the first 10 characters:

❍ SBSI0200 must be specified for the format name parameter.

❍ the value in the second 10 characters must be blank.

The second 10 characters contain the library name. You can use one of these special values for the library name:

*CURLIB   The job's current library

*LIBL   The library list

If *ACTIVE is specified in the first 10 characters, the value in the second 10 characters must be blank.

If information for a subsystem description is requested more than once in the array, the receiver variable will not contain duplicate information for that subsystem description.

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

Number of qualified subsystem names

>   INPUT; BINARY(4)

>   The number of names specified in the qualified subsystem name parameter. If this parameter is not specified, the number of names specified defaults to 1. If this parameter is specified, the number specified must be in the range 1 to 65535. If a number greater than 1 is specified, SBSI0200 must be specified for the format name parameter.

SBSI0100 Format

The following table shows the information returned in the receiving variable for the SBSI0100 format. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Subsystem description name
18	12	CHAR(10)	Subsystem description library name
28	1C	CHAR(10)	Subsystem status
38	26	CHAR(10)	Sign-on device file name
48	30	CHAR(10)	Sign-on device file library name
58	3A	CHAR(10)	Secondary language library name
68	44	BINARY(4)	Maximum active jobs
72	48	BINARY(4)	Currently active jobs
76	4C	BINARY(4)	Number of storage pools defined
Offsets vary. These five fields repeat, in the order listed, for each pool defined for the subsystem.		BINARY(4)	Pool ID
		CHAR(10)	Pool name
		CHAR(6)	Reserved
		BINARY(4)	Pool size
		BINARY(4)	Pool activity level

SBSI0200 Format

The following table shows the information returned in the receiving variable for the SBSI0200 format. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available

8	8	BINARY(4)	Offset to first subsystem entry
12	C	BINARY(4)	Number of subsystem entries returned
16	10	BINARY(4)	Size of a subsystem entry
20	14	CHAR(*)	Reserved
Offsets vary. These fields repeat, in the order listed, for each subsystem.		CHAR(10)	Subsystem description name
		CHAR(10)	Subsystem description library name
		CHAR(12)	Subsystem extended status
		BINARY(4)	Maximum active jobs
		BINARY(4)	Currently active jobs
		CHAR(10)	Subsystem monitor job name
		CHAR(10)	Subsystem monitor job user
		CHAR(6)	Subsystem monitor job number
		CHAR(50)	Subsystem description text
		CHAR(*)	Reserved

Field Descriptions

Bytes available. The total length of all data available.

Bytes returned. The length of the data actually returned. The number of bytes returned is always less than or equal to both the number of bytes available and the receiving variable length.

Currently active jobs. The number of jobs currently active in the subsystem. This number *includes* held jobs but *excludes* jobs that are disconnected or suspended because of a transfer secondary job or a transfer group job. If the subsystem status is *INACTIVE, this number is 0.

Maximum active jobs. The maximum number of jobs that can run or use resources in the subsystem at one time. If the subsystem description specifies *NOMAX, indicating that there is no maximum, this number is -1.

Number of subsystem entries returned. The number of subsystems for which entries are returned. This number determines how many times the entire set of fields describing a subsystem is repeated.

Number of storage pools defined. The number of storage pools defined for the subsystem. The maximum number of storage pools for a subsystem is currently 10. This number determines how many times the pool ID, pool name, pool size, pool activity level, and reserved fields are repeated. Those five fields are repeated as a group for each pool defined for the subsystem.

Offset to first subsystem entry. The number of bytes from the first byte of the receiver variable to the information for the first subsystem.

Pool activity level. The maximum number of jobs that can be active in the pool at one time. If the pool name indicates a system-defined pool, the number returned is 0.

Pool ID. The pool ID for the subsystem pool.

Pool name. The name of the subsystem pool. If the pool is user-defined, the value of this field is *USERPOOL. If the pool is system-defined, the value is one of these names:

BASE	The system base pool, which can be shared with other subsystems. The QBASPOOL system value defines the base pool's minimum size. The base pool contains all main storage not allocated to other pools. The QBASACTLVL system value defines its activity level.
INTERACT	The shared pool used for interactive work.
NOSTG	No storage size or activity level is assigned to this storage pool.
*SHRPOOL1-*SHRPOOL60*	Shared pools.
SPOOL	The shared pool for spooling writers.

The Change Shared Storage Pool (CHGSHRPOOL) command specifies the size and activity level of shared pools.

Pool size. If the pool name is *USERPOOL, the amount of storage, in kilobytes, that the pool attempts to allocate. If the pool has any other name, the value of this field is 0.

Reserved. An ignored field.

Secondary language library name. The name of the subsystem's secondary language library.

Sign-on device file library name. The name of the library in which the sign-on device file resides.

Size of a subsystem entry. The number of bytes in the entry for a subsystem.

Subsystem description library name. The name of the library in which the subsystem description resides.

Subsystem description name. The name of the subsystem about which information is being returned.

Subsystem description text. The text description of the subsystem description. The field is blank if there is no text description.

Subsystem extended status. Possible values that can be returned for subsystem extended status are:

ACTIVE	The subsystem is running.
ENDING	An ENDSBS command has been issued for the subsystem or an ENDSYS command has been issued, but the subsystem is still running.
INACTIVE	The subsystem is not running.
RESTRICTED	An ENDSBS command for the controlling subsystem, an ENDSYS *ALL command, or an ENDSYS command has placed the controlling subsystem in a restricted condition.
STARTING	A STRSBS command has been issued for the subsystem, but it is still in the process of being started.

Subsystem monitor job name. The name for the subsystem monitor job as identified to the system. The field is blank if the subsystem extended status field is *INACTIVE.

Subsystem monitor job number. The system-assigned number for the subsystem monitor job. The field is blank if the subsystem extended status field is *INACTIVE.

Subsystem monitor job user. The name of the user profile under which the subsystem monitor job is running. The field is blank if the subsystem extended status field is *INACTIVE.

Subsystem status. Possible values that can be returned for subsystem status are:

ACTIVE The subsystem is running.

INACTIVE The subsystem is not running.

Error Messages

Message ID	Error Message Text
CPF1605 E	Cannot allocate subsystem description &1.
CPF1606 E	Error during allocation of subsystem &1.
CPF1607 E	Previous request pending for subsystem &1.
CPF1608 E	Subsystem description &1 not found.
CPF1619 E	Subsystem description &1 in library &2 damaged.
CPF1835 E	Not authorized to subsystem description.
CPF187A E	List of active subsystems not available.
CPF1877 E	Incorrect format specified.
CPF1878 E	Library name not valid for subsystem &1.
CPF3CF1 E	Error code parameter not valid.
» CPF3CF2 E	Error(s) occurred during running of &1 API.«
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8122 E	&8 damage on library &4.
CPF9807 E	One or more libraries in library list deleted.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Retrieve System Status (QWCRSSTS) API

> Required Parameter Group:
>
> | | | | |
> |---|---|---|---|
> | 1 | Receiver variable | Output | Char(*) |
> | 2 | Length of receiver variable | Input | Binary(4) |
> | 3 | Format name | Input | Char(8) |
> | 4 | Reset status statistics | Input | Char(10) |
> | 5 | Error Code | I/O | Char(*) |
>
> Optional Parameter Group:
>
> | | | | |
> |---|---|---|---|
> | 6 | Pool selection information | Input | Char(*) |
> | 7 | Size of pool selection information | Input | Binary(4) |
>
> Default Public Authority: *USE
>
> Threadsafe: No

The Retrieve System Status (QWCRSSTS) API allows you to retrieve a group of statistics that represents the current status of the system.

Authorities and Locks

None.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that will receive the system status information being retrieved. For the format, see [Format of Data Returned](#).

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable described in [Format of Data Returned](#). If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The format of the information to be returned. You must use one of the following format names:

SSTS0100   Basic system status information about the signed-on users and batch jobs in the system. The information returned in this format is similar to the basic display of the Display System Status (DSPSYSSTS) command.

SSTS0200   System status information. The information returned in this format is similar to the disk information of the intermediate or advanced display of the DSPSYSSTS command.

SSTS0300   System status information. The information returned in this format is similar to the pool information of the intermediate or advanced display of the DSPSYSSTS command.

SSTS0400   Pool status information. The information returned in this format is the SSTS0300 format information and additional pool information. The pool selection information parameter must be used when this format is used.

SSTS0500   Pool subsystem information. The information returned in this format is a list of active subsystems using a pool. The pool selection information parameter must be used to select one system pool when this format is used.

For more information about these formats, see [Format of Data Returned](#).

Reset status statistics

INPUT; CHAR(10)

Whether the status statistics and elapsed time are reset to zero, as if this were the first call to the API. The statistics will be reset before new information is gathered. This parameter will also reset the status statistics on the DSPSYSSTS and Work with System Status (WRKSYSSTS) commands. This parameter is ignored for format SSTS0100 and format SSTS0500.

*YES   Statistics will be reset to zero.

*NO   Statistics will not be reset to zero.

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.

Optional Parameter Group

Pool selection information

INPUT; CHAR(*)

Information that is used for selecting which pools to include in the list. This parameter only applies to the SSTS0400 and SSTS0500 formats. See [Format of Pool Selection Information](#) for a description on the layout of this parameter. If this optional parameter is specified, the Size of pool selection information parameter must also be specified.

Size of pool selection information

> INPUT; BINARY(4)

> The size, in bytes, of the pool selection information parameter. If the value of this parameter is 0, the pool selection information is not used. The valid values for this parameter are 0, 20, or 24. If this parameter is not specified, the Size of pool selection information is defaulted to 0.

Format of Data Returned

The receiver variable holds the system status information returned.

SSTS0100 Format

The following table shows the information returned for the SSTS0100 format. For a detailed description of each field see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes available
4	4	BINARY(4)	Bytes returned
8	8	CHAR(8)	Current date and time
16	10	CHAR(8)	System name
24	18	BINARY(4)	Users currently signed on
28	1C	BINARY(4)	Users temporarily signed off (disconnected)
32	20	BINARY(4)	Users suspended by system request
36	24	BINARY(4)	Users suspended by group jobs
40	28	BINARY(4)	Users signed off with printer output waiting to print
44	2C	BINARY(4)	Batch jobs waiting for messages
48	30	BINARY(4)	Batch jobs running
52	34	BINARY(4)	Batch jobs held while running
56	38	BINARY(4)	Batch jobs ending
60	3C	BINARY(4)	Batch jobs waiting to run or already scheduled
64	40	BINARY(4)	Batch jobs held on a job queue
68	44	BINARY(4)	Batch jobs on a held job queue
72	48	BINARY(4)	Batch jobs on an unassigned job queue
76	4C	BINARY(4)	Batch jobs ended with printer output waiting to print

SSTS0200 Format

The following table shows the information returned for the SSTS0200 format. For a detailed description of each field, see the [Field Descriptions](#).

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes available
4	4	BINARY(4)	Bytes returned
8	8	CHAR(8)	Current date and time
16	10	CHAR(8)	System name
24	18	CHAR(6)	Elapsed time
30	1E	CHAR(1)	Restricted state flag
31	1F	CHAR(1)	Reserved
32	20	BINARY(4)	% processing unit used
36	24	BINARY(4)	Jobs in system
40	28	BINARY(4)	% permanent addresses
44	2C	BINARY(4)	% temporary addresses
48	30	BINARY(4)	System ASP
52	34	BINARY(4)	% system ASP used
56	38	BINARY(4)	Total auxiliary storage
60	3C	BINARY(4)	Current unprotected storage used
64	40	BINARY(4)	Maximum unprotected storage used
68	44	BINARY(4)	% DB capability
72	48	BINARY(4)	Main storage size
≫ 76	4C	BINARY(4)	Number of partitions
80	50	BINARY(4)	Partition identifier
84	54	BINARY(4)	Reserved
88	58	BINARY(4)	Current processing capacity
92	5C	CHAR(1)	Processor sharing attribute
93	5D	CHAR(3)	Reserved
96	60	BINARY(4)	Number of processors
100	64	BINARY(4)	Active jobs in system
104	68	BINARY(4)	Active threads in system
108	6C	BINARY(4)	Maximum jobs in system
112	70	BINARY(4)	% temporary 256MB segments used
116	74	BINARY(4)	% temporary 4GB segments used
120	78	BINARY(4)	% permanent 256MB segments used
124	7C	BINARY(4)	% permanent 4GB segments used
128	80	BINARY(4)	% current interactive performance ≪

SSTS0300 Format

The following table shows the information returned for the SSTS0300 format. For a detailed description of each field, see the [Field Descriptions](#).

Offset Dec	Offset Hex	Type	Field
0	0	BINARY(4)	Bytes available
4	4	BINARY(4)	Bytes returned
8	8	CHAR(8)	Current date and time
16	10	CHAR(8)	System name
24	18	CHAR(6)	Elapsed time
30	1E	CHAR(2)	Reserved
32	20	BINARY(4)	Number of pools
36	24	BINARY(4)	Offset to pool information
40	28	BINARY(4)	Length of pool information entry
44	2C	CHAR(*)	Reserved
Offsets vary. These fields repeat, in the order listed, for each pool allocated by the system.		BINARY(4)	System pool
		BINARY(4)	Pool size
		BINARY(4)	Reserved size
		BINARY(4)	Maximum active threads
		BINARY(4)	Database faults
		BINARY(4)	Database pages
		BINARY(4)	Nondatabase faults
		BINARY(4)	Nondatabase pages
		BINARY(4)	Active to wait
		BINARY(4)	Wait to ineligible
		BINARY(4)	Active to ineligible
		CHAR(10)	Pool name
		CHAR(10)	Subsystem name
		CHAR(10)	Subsystem library name
		CHAR(10)	Paging option

SSTS0400 Format

The following table shows the information returned for the SSTS0400 format. For a detailed description of each field, see the [Field Descriptions](#).

Offset Dec	Offset Hex	Type	Field
0	0	BINARY(4)	Bytes available
4	4	BINARY(4)	Bytes returned
8	8	CHAR(8)	Current date and time

16	10	CHAR(8)	System name
24	18	CHAR(6)	Elapsed time
30	1E	CHAR(2)	Reserved
32	20	BINARY(4)	Main storage size
36	24	BINARY(4)	Minimum machine pool size
40	28	BINARY(4)	Minimum base pool size
44	2C	BINARY(4)	Number of pools
48	30	BINARY(4)	Offset to pool information
52	34	BINARY(4)	Length of pool information entry
56	38	CHAR(*)	Reserved
Note: The following fields repeat, in the order listed, for the number of pools returned.			
		BINARY(4)	System pool
		BINARY(4)	Pool size
		BINARY(4)	Reserved size
		BINARY(4)	Maximum active threads
		BINARY(4)	Database faults
		BINARY(4)	Database pages
		BINARY(4)	Nondatabase faults
		BINARY(4)	Nondatabase pages
		BINARY(4)	Active to wait
		BINARY(4)	Wait to ineligible
		BINARY(4)	Active to ineligible
		CHAR(10)	Pool name
		CHAR(10)	Subsystem name
		CHAR(10)	Subsystem library name
		CHAR(10)	Paging option
		BINARY(4)	Defined size
		BINARY(4)	Current threads
		BINARY(4)	Current ineligible threads
		BINARY(4)	Tuning priority
		BINARY(4)	Tuning minimum pool size %
		BINARY(4)	Tuning maximum pool size %
		BINARY(4)	Tuning minimum faults
		BINARY(4)	Tuning per-thread faults
		BINARY(4)	Tuning maximum faults
		CHAR(50)	Description
		CHAR(1)	Status
		CHAR(*)	Reserved

SSTS0500 Format

The following table shows the information returned for the SSTS0500 format. For a detailed description of each field, see the Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes available
4	4	BINARY(4)	Bytes returned
8	8	CHAR(8)	Current date and time
16	10	CHAR(8)	System name
24	18	BINARY(4)	System pool
28	1C	BINARY(4)	Number of subsystems available
32	20	BINARY(4)	Number of subsystems returned
36	24	BINARY(4)	Offset to subsystem information
40	28	BINARY(4)	Length of subsystem information entry
44	32	CHAR(10)	Pool name
54	36	CHAR(*)	Reserved
Note: The following fields repeat, in the order listed, for the number of subsystems returned.			
		CHAR(10)	Subsystem name
		CHAR(10)	Subsystem library name
		CHAR(*)	Reserved

Field Descriptions

≫% current interactive performance. The percentage of interactive performance assigned to this logical partition. This value is a percentage of the total interactive performance available to the entire physical system. For example, a value of 41 in binary would be 41 percent.≪

% DB capability. The percentage of processor database capability (in tenths) that was used during the elapsed time. Database capability is the maximum CPU utilization available for database processing on this server. -1 is returned if this server does not report the amount of CPU used for database processing. For example, a value of 411 in binary would be 41.1 percent.

% permanent addresses. The percentage (in thousandths) of the maximum possible addresses for permanent objects that have been used. For example, a value of 41123 in binary would be 41.123 percent.

≫% permanent 256MB segments used. The percentage (in thousandths) of the maximum possible permanent 256MB segments that have been used. For example, a value of 41123 in binary would be 41.123 percent.

% permanent 4GB segments used. The percentage (in thousandths) of the maximum possible permanent 4GB segements that have been used. For example, a value of 41123 in binary would be 41.123 percent.≪

% processing unit used. The average (in tenths) of the elapsed time during which the processing units were in use. For example, a value of 411 in binary would be 41.1 percent.

% system ASP used. The percentage (in ten thousandths) of the system storage pool currently in use. For example, a value of 41123 in binary would be 4.1123 percent.

% temporary addresses. The percentage (in thousandths) of the maximum possible addresses for temporary objects that have been used. For example, a value of 41123 in binary would be 41.123 percent.

≫% temporary 256MB segments used. The percentage (in thousandths) of the maximum possible temporary 256MB segments that have been used. For example, a value of 41123 in binary would be 41.123 percent.

% temporary 4GB segments used. The percentage (in thousandths) of the maximum possible temporary 4GB segments that have been used. For example, a value of 41123 in binary would be 41.123 percent.

Active jobs in system. The number of jobs active in the system (jobs that have been started, but have not yet ended), including both user and system jobs.

Active threads in system. The number of initial and secondary threads in the system (threads that have been started, but have not yet ended), including both user and system threads.≪

Active to ineligible. The rate (in tenths), in transitions per minute, of transitions of threads from an active condition to an ineligible condition. For example, a value of 123 in binary would be 12.3 transitions per minute.

Active to wait. The rate (in tenths), in transitions per minute, of transitions of threads from an active condition to a waiting condition. For example, a value of 123 in binary would be 12.3 transitions per minute.

Batch jobs ended with printer output waiting to print. The number of completed batch jobs that produced printer output that is waiting to print.

Batch jobs ending. The number of batch jobs that are in the process of ending due to one of the following conditions:

- The job finishes processing normally.
- The job ends before its normal completion point and is being removed from the system.

Batch jobs held on a job queue. The number of batch jobs that were submitted, but were held before they could begin running.

Batch jobs held while running. The number of batch jobs that had started running, but are now held.

Batch jobs on a held job queue. The number of batch jobs on job queues that have been assigned to a subsystem, but are being held.

Batch jobs on an unassigned job queue. The number of batch jobs on job queues that have not been assigned to a subsystem.

Batch jobs running. The number of batch jobs currently running on the system.

Batch jobs waiting for messages. The number of batch jobs waiting for a reply to a message before they can continue to run.

Batch jobs waiting to run or already scheduled. The number of batch jobs on the system that are currently waiting to run, including those that were submitted to run at a future date and time. Jobs on the job schedule that have not been submitted are not included.

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned. The number of bytes returned is always less than or equal to both the number of bytes available and the receiving variable length.

Current date and time. The date and time when the status was gathered. This is in system timestamp format.

Current ineligible threads. The number of ineligible threads in the pool's activity level.

≫Current processing capacity. The amount (in hundredths) of current processing capacity of the partition. For a partition sharing physical processors, this attribute represents the share of the physical processors in the pool it is executing. For example, a value of 233 means that the partition's current processing capacity is equivalent to 2.33 physical processors. If the current number of processors in the partition is four, each virtual processor has 0.58 the computing capacity of a physical processor in the physical machine. For a partition using dedicated processors, the value represents the number of virtual processors that are currently active in the partition. For example, a partition using four dedicated processors will return a value of 400 for the current processing capacity.≪

Current threads. The number of threads currently using the pool's activity level.

Current unprotected storage used. The current amount of storage in use for temporary objects. This value is in millions (M) of bytes.

Database faults. The rate (in tenths), shown in page faults per second, of database page faults against pages containing either database data or access paths. A page fault is a program notification that occurs when a page that is marked as not in main storage is referred to by an active program. An access path is the means by which the system provides a logical organization to the data in a database file. For example, a value of 123 in binary would be 12.3 page faults per second.

Database pages. The rate (in tenths), in pages per second, at which database pages are brought into the storage pool. A page is a 4096-byte block of information that is transferable between auxiliary storage and main storage. For example, a value of 123 in binary would be 12.3 pages per second.

Defined size. The size of the pool, in kilobytes, as defined in the shared pool, subsystem description, or system value QMCHPOOL. -1 will be returned for pools without a defined size.

Description. The description of the shared pool. This field is blank for private pools defined in subsystem descriptions.

Elapsed time. The time that has elapsed between the measurement start time and the current system time. This value is in the format HHMMSS where HH is the hour, MM is the minute, and SS is the second.

Jobs in system. The total number of user jobs and system jobs that are currently in the system. The total includes:

- All jobs on job queues waiting to be processed.
- All jobs currently active (being processed).
- All jobs that have completed running but still have output on output queues to be produced.

Length of pool information entry. The length of the information returned for each pool. If the receiver variable was not sufficiently large to hold all of the pool information, the amount of pool information returned may be less than this value.

Length of subsystem information entry. The length of the information returned for each subsystem. If the receiver variable was not sufficiently large to hold all of the information, the amount of information returned may be less than this value.

Main storage size. The amount of main storage, in kilobytes, in the system. On a partitioned system, the

main storage size can change while the system is active.

Maximum active threads. The maximum number of threads that can be active in the pool at any one time.

»Maximum jobs in system. The maximum number of jobs that are allowed on the system. When the number of jobs reaches this maximum, you can no longer submit or start more jobs on the system. The total includes:

- All jobs on job queues waiting to be processed.
- All jobs currently active (being processed).
- All jobs that have completed running but still have output on output queues to be produced.«

Maximum unprotected storage used. The largest amount of storage for temporary objects used at any one time since the last IPL. This value is in millions (M) of bytes.

Minimum base pool size. The minimum size, in kilobytes, for the base pool. This value is the QBASPOOL system value.

Minimum machine pool size. The minimum size, in kilobytes, for the machine pool. This value is the minimum size required by the machine for the machine pool.

Nondatabase faults. The rate (in tenths), in page faults per second, of nondatabase page faults against pages other than those designated as database pages. For example, a value of 123 in binary would be 12.3 page faults per second.

Nondatabase pages. The rate (in tenths), in pages per second, at which nondatabase pages are brought into the storage pool. For example, a value of 123 in binary would be 12.3 pages per second.

»Number of partitions. The number of partitions on the system. This includes partitions that are currently powered on (running) and partitions that are powered off.«

Number of pools. The number of pools allocated when the information was gathered. This number may be larger than the number of pools information is returned for if the receiver variable is not large enough.

»Number of processors. The number of processors that are currently active in this partition.«

Number of subsystems available. The number of subsystems using the pool.

Number of subsystems returned. The number of subsystems returned in the receiver variable.

Offset to pool information. The offset from the beginning of the structure to the start of the pool information.

Offset to subsystem information. The offset from the beginning of the structure to the start of the subsystem information.

Paging option. Whether the system will dynamically adjust the paging characteristics of the storage pool for optimum performance. The following special values may be returned.

FIXED   The system does not dynamically adjust the paging characteristics.

CALC   The system dynamically adjusts the paging characteristics.

USRDFN   The system does not dynamically adjust the paging characteristics for the storage pool but uses values that have been defined through an API.

»Partition identifier. The identifier for the current partition in which the API is running.«

Pool name. The name of this storage pool. The name may be a number, in which case it is a private pool associated with a subsystem. The following special values may be returned.

*MACHINE	The specified pool definition is defined to be the machine pool.
*BASE	The specified pool definition is defined to be the base system pool, which can be shared with other subsystems.
*INTERACT	The specified pool definition is defined to be the shared pool used for interactive work.
*SPOOL	The specified pool definition is defined to be the shared pool used for spooled writers.
*SHRPOOL1- *SHRPOOL60	The specified pool definition is defined to be a shared pool.

Pool size. The amount of main storage, in kilobytes, in the pool.

»Processor sharing attribute. This attribute indicates whether this partition is sharing processors on the current IPL of this partition. If the value indicates the partition does not share physical processors, then this partition uses only dedicated processors. If the value indicates the partition shares physical processors, then this partition uses physical processors from a shared pool of physical processors. The following values are returned:

0   Partition does not share processors

1   Partition shares processors«

Reserved. An ignored field.

Reserved size. The amount of storage, in kilobytes, in the pool reserved for system use (for example, for save/restore operations). The system calculates this amount by using storage pool sizes and activity levels.

Restricted state flag. Whether the system is in restricted state. The following values are returned:

0   System is not in restricted state.

1   System is in restricted state.

Status. The status of the pool:

0   Active

1   Inactive

Subsystem library name. The library containing the subsystem description. This field will be blank for shared pools (formats SSTS0300 and SSTS0400).

Subsystem name. The subsystem with which this storage pool is associated. This field will be blank for shared pools (formats SSTS0300 and SSTS0400).

System ASP. The storage capacity of the system auxiliary storage pool (ASP1). This value is in millions (M) of bytes.

System name. The name of the system where the statistics were collected.

System pool. The system-related pool identifier for each of the system storage pools that currently has

main storage allocated to it.

Total auxiliary storage. The total auxiliary storage (in millions of bytes) on the system.

Tuning maximum faults. The maximum page faults per second (in hundredths) to use as a guideline for the shared storage pool. For example, a value of 1234 would be 12.34 page faults per second. This field is 0 for private pools defined in subsystem descriptions.

Tuning maximum pool size %. The maximum amount of storage to allocate to the shared storage pool (as a percentage of total main storage). The value returned is in hundredths. For example, a value of 1234 would be 12.34 percent. This field is 0 for private pools defined in subsystem descriptions.

Tuning minimum faults. The minimum page faults per second (in hundredths) to use as a guideline for the shared storage pool. For example, a value of 1234 would be 12.34 page faults per second. This field is 0 for private pools defined in subsystem descriptions.

Tuning minimum pool size %. The minimum amount of storage to allocate to the shared storage pool (as a percentage of total main storage). The value returned is in hundredths. For example, a value of 1234 would be 12.34 percent. This field is 0 for private pools defined in subsystem descriptions.

Tuning per-thread faults. The page faults per second (in hundredths) for each active thread to use as a guideline for the shared storage pool. For example, a value of 1234 would be 12.34 page faults per second. This field is 0 for private pools defined in subsystem descriptions.

Tuning priority. The priority of the shared storage pool used by the system when making automatic performance adjustments. This field is 0 for private pools defined in subsystem descriptions.

Users currently signed on. The number of users currently signed on the system. System request jobs and group jobs are not included in this number.

Users signed off with printer output waiting to print. The number of sessions that have ended with printer output files waiting to print.

Users suspended by group jobs. The number of user jobs that have been temporarily suspended by group jobs so that another job may be run.

Users suspended by system request. The number of user jobs that have been temporarily suspended by system request jobs so that another job may be run.

Users temporarily signed off (disconnected). The number of jobs that have been disconnected due to either the selection of option 80 (Temporary sign-off) or the entry of the Disconnect Job (DSCJOB) command.

Wait to ineligible. The rate (in tenths), in transitions per minute, of transitions of threads from a waiting condition to an ineligible condition. For example, a value of 123 in binary would be 12.3 transitions per minute.

Format of Pool Selection Information

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Type of pool
10	A	CHAR(10)	Shared pool name
20	14	BINARY(4)	System pool identifier

Selection Field Descriptions

Shared pool name. This is used to select a shared pool when type of pool is *SHARED. The possible values are:

*ALL	All shared pools are returned.
*MACHINE	The machine pool is returned.
*BASE	The base pool is returned.
*INTERACT	The shared pool used for interactive work is returned.
*SPOOL	The shared pool used for spool writers is returned.
*SHRPOOL1-60	The specified shared pool is returned.

When *ALL is specified the pools are returned in the order *MACHINE, *BASE, *INTERACT, *SPOOL, and *SHRPOOL1-60. This field must be blank when *SYSTEM is specified for type of pool.

System pool identifier. This is used to select an active system pool when type of pool is *SYSTEM. The possible values are:

-1	All active pools are returned.
1 - 64	The specified active pool is returned. If the pool is not active, CPF186B is sent.

When -1 is specified only the active pools are returned in system pool identifier order. This field must be 0 when *SHARED is specified for type of pool.

Type of pool. The type of pools to include in the list. The possible special values follow:

*SHARED	The shared pools identified by the shared pool name field in the selection information. The other selection fields are not used. The selection information size must be a minimum of 20.
*SYSTEM	The system pools identified by the system pool identifier field in the selection information. The other selection fields are not used. The selection information size must be a minimum of 24.
	The pool selection information parameter only applies to the SSTS0400 and SSTS0500 formats. When other formats are used, the size of pool selection information parameter must be 0 or not specified.

Error Messages

Message ID	Error Message Text
CPF1E99 E	Unexpected error occurred.

CPF186A E	Selection information not allowed with format &1.
CPF186B E	Pool &1 not active.
CPF186C E	Selection information required with format &1.
CPF1869 E	Value &1 for reset status statistics not valid.
CPF187A E	List of active subsystems not available.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Retrieve System Values (QWCRSVAL) API

```
Required Parameter Group:

    1    Receiver variable                    Output      Char(*)
    2    Length of receiver variable          Input       Binary(4)
    3    Number of system values to retrieve  Input       Binary(4)
    4    System value names                   Input       Array(*) of Char(10)
    5    Error code                           I/O         Char(*)

Default Public Authority: *USE

Threadsafe: Yes
```

The Retrieve System Values (QWCRSVAL) API lets you retrieve system values.

Authorities and Locks

Job Authority

      No special authority is required to use this API.

Required Parameter Group

Receiver variable

      OUTPUT; CHAR(*)

      The variable that is to receive the information requested. For the format, see Format of Data Returned.

Length of receiver variable

      INPUT; BINARY(4)

      The length of the receiver variable described in the Format of Data Returned. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 28 bytes.

Number of system values to retrieve

      INPUT; BINARY(4)

      The total number of system values to retrieve.

System value names

      INPUT; ARRAY(*) of CHAR(10)

      The names of the system values to be retrieved. This can be a list of system value names where each name is 10 characters.

Error code

     I/O; CHAR(*)

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

Format of Data Returned

The receiver variable holds the information returned about each system value.

The receiver variable has three logical parts:

1. The first field specifies the number of system values returned.
2. The next fields give the offsets to the system values returned. There is one offset field for each system value returned.
3. Next are the system value information tables for the system values returned. There is one system value information table for each system value.

The following table shows the format of the receiver variable. The offset fields are repeated until the offsets for all the system values returned are listed; the system value information table for each system value is repeated in the same way. For a detailed description of each field, see the Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of system values returned
4	4	ARRAY(*) of BINARY(4)	Offset to system value information table
*	*	CHAR(*)	System value information table. This field is repeated for each system value returned.
Note: Each system value in the table is represented by the standard system value information table described in System Value Information Table.			

To determine the length of the receiver variable, the following calculation should be done. For each system value to be returned, get the length of the data returned for the system value and add 24. After adding the lengths for each system value, add 4. This calculation takes into account the data alignment that needs to be done; therefore, this value is a worst-case estimate.

System Value Information Table

The following table shows the format of the system value information table.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(10)	System value
10	A	CHAR(1)	Type of data
11	B	CHAR(1)	Information status
12	C	BINARY(4)	Length of data
16	10	CHAR(*)	Data

Field Descriptions

Data. The data returned for the system value.

Information status. Whether the information was available for the system value.

blank   The information was available.

L   The information was not available because the system value was locked.

Offset to system values information table. The offset from the beginning of the structure to the start of the system value information.

Length of data. The length of the data returned for the system value. If the information was not available, the length will be zero.

System value. The system value to be retrieved. See Valid System Values for the list of valid system values.

Number of system values returned. The number of system values returned to the application.

Type of data. The type of data returned.

C   The data is returned in character format.

B   The data is returned in binary format.

blank   The data is not available.

Valid System Values

For a detailed description of each field, see System Value Field Descriptions.

System value	Type	Description
QABNORMSW	CHAR(1)	Previous end of system indicator
QACGLVL	ARRAY(8) of CHAR(10)	Accounting level
QACTJOB	BINARY(4)	Active jobs
QADLACTJ	BINARY(4)	Additional active jobs
QADLSPLA	BINARY(4)	Additional storage
QADLTOTJ	BINARY(4)	Additional total jobs
QALWOBJRST	ARRAY(15) of CHAR(10)	Allow object restore options
QALWUSRDMN	ARRAY(50) of CHAR(10)	Allow user domain
QASTLVL	CHAR(10)	Assistance level
QATNPGM	CHAR(20)	Attention program
QAUDCTL	ARRAY(5) of CHAR(10)	Auditing control

QAUDENDACN	CHAR(10)	Auditing end action
QAUDFRCLVL	BINARY(4)	Auditing force level
QAUDLVL	ARRAY(16) of CHAR(10)	Auditing level
QAUTOCFG	CHAR(1)	Automatic configuration indicator
QAUTORMT	CHAR(1)	Automatic configuration for remote controllers
QAUTOSPRPT	CHAR(1)	Automatic system disabled reporting
QAUTOVRT	BINARY(4)	Automatic configuration for virtual devices
QBASACTLVL	BINARY(4)	Base activity level
QBASPOOL	BINARY(4)	Base pool minimum size
QBOOKPATH	ARRAY(5) of CHAR(63)	Book and bookshelf search path
QCCSID	BINARY(4)	Coded character set identifier
QCENTURY	CHAR(1)	Century indicator
QCFGMSGQ	CHAR(20)	Configuration message queue
QCHRID	CHAR(20)	Character set and code page
QCHRIDCTL	CHAR(10)	Character identifier control
QCMNARB	CHAR(10)	Communication arbiters
QCMNRCYLMT	CHAR(20)	Communications recovery limit
QCNTRYID	CHAR(2)	Country or region identifier
QCONSOLE	CHAR(10)	Console name
QCRTAUT	CHAR(10)	Create authority
QCRTOBJAUD	CHAR(10)	Create object auditing
QCTLSBSD	CHAR(20)	Controlling subsystem
QCURSYM	CHAR(1)	Currency symbol
QDATE	CHAR(7)	System date
QDATFMT	CHAR(3)	Date format
QDATSEP	CHAR(1)	Date separator
QDAY	CHAR(3)	Day
QDAYOFWEEK	CHAR(4)	Day of the week
» QDBFSTCCOL	CHAR(10)	Database file statistics collection «
QDBRCVYWT	CHAR(1)	Database recovery wait
QDECFMT	CHAR(1)	Decimal format
QDEVNAMING	CHAR(10)	Device naming convention
QDEVRCYACN	CHAR(20)	Device recovery action
QDSCJOBITV	CHAR(10)	Disconnect job interval
QDSPSGNINF	CHAR(1)	Sign-on information
QDYNPTYADJ	CHAR(1)	Dynamic priority adjustment
QDYNPTYSCD	CHAR(1)	Dynamic priority scheduler
QFRCCVNRST	CHAR(1)	Force conversion on restore
QHOUR	CHAR(2)	Hour
QHSTLOGSIZ	BINARY(4)	History log size
QIGC	CHAR(1)	DBCS installed

QIGCCDEFNT	CHAR(20)	Double-byte coded font name
QIGCFNTSIZ	BINARY(4)	Double-byte coded font point size
QINACTITV	CHAR(10)	Inactive job time-out
QINACTMSGQ	CHAR(20)	Inactive message queue
QIPLDATTIM	CHAR(13)	Automatic IPL date and time
QIPLSTS	CHAR(1)	IPL status
QIPLTYPE	CHAR(1)	IPL type
QJOBMSGQFL	CHAR(10)	Job message queue full
QJOBMSGQMX	BINARY(4)	Job message queue maximum size
QJOBMSGQSZ	BINARY(4)	Job message queue initial size
QJOBMSGQTL	BINARY(4)	Maximum job message queue initial size
QJOBSPLA	BINARY(4)	Initial spooling size
QKBDBUF	CHAR(10)	Keyboard buffer
QKBDTYPE	CHAR(3)	Keyboard type
QLANGID	CHAR(3)	Language identifier
QLEAPADJ	BINARY(4)	Leap year adjustment
QLIBLCKLVL	CHAR(1)	Library locking level
QLMTDEVSSN	CHAR(1)	Limit device session
QLMTSECOFR	CHAR(1)	Limit security officer
QLOCALE	CHAR(2080)	Locale path name
QMAXACTLVL	BINARY(4)	Maximum activity level
QMAXJOB	BINARY(4)	Maximum number of jobs
QMAXSGNACN	CHAR(1)	Maximum sign-on action
QMAXSIGN	CHAR(6)	Maximum not valid sign-on
QMAXSPLF	BINARY(4)	Maximum spooled files per job
QMCHPOOL	BINARY(4)	Machine pool size
QMINUTE	CHAR(2)	Minute
QMLTTHDACN	CHAR(1)	Multithreaded job action
QMODEL	CHAR(4)	System model
QMONTH	CHAR(2)	Month
QPASTHRSVR	CHAR(10)	Pass-through servers
QPFRADJ	CHAR(1)	Performance adjustment
QPRBFTR	CHAR(20)	Problem filter
QPRBHLDITV	BINARY(4)	Problem hold interval
QPRCMLTTSK	CHAR(1)	Processor multitasking
QPRCFEAT	CHAR(4)	Processor feature code
QPRTDEV	CHAR(10)	Printer device
QPRTKEYFMT	CHAR(10)	Print key format
QPRTTXT	CHAR(30)	Print text
QPWDEXPITV	CHAR(6)	Days password valid
QPWDLMTAJC	CHAR(1)	Limit adjacent digits
QPWDLMTCHR	CHAR(10)	Limit characters

QPWDLMTREP	CHAR(1)	Limit repeat characters
QPWDLVL	BINARY(4)	Password level
QPWDMAXLEN	BINARY(4)	Maximum password length
QPWDMINLEN	BINARY(4)	Minimum password length
QPWDPOSDIF	CHAR(1)	Limit character positions
QPWDRQDDGT	CHAR(1)	Required password digits
QPWDRQDDIF	CHAR(1)	Duplicate password
QPWDVLDPGM	CHAR(20)	Password validation program
QPWRDWNLMT	BINARY(4)	Power down limit
QPWRRSTIPL	CHAR(1)	Power restore IPL
QQRYDEGREE	CHAR(10)	Parallel processing degree
QQRYTIMLMT	CHAR(10)	Query processing time limit
QRCLSPLSTG	CHAR(10)	Reclaim spool storage
QRETSVRSEC	CHAR(1)	Retain server security data
QRMTIPL	CHAR(1)	Remote IPL
QRMTSRVATR	CHAR(1)	Remote service attribute
QRMTSIGN	CHAR(20)	Remote sign-on
QSCPFCONS	CHAR(1)	IPL action with console problem
QSECOND	CHAR(2)	Second
QSECURITY	CHAR(2)	Security level
QSETJOBATR	ARRAY(16) of CHAR(10)	Set job attributes from locale
QSFWERRLOG	CHAR(10)	Software error log
QSHRMEMCTL	CHAR(1)	Shared memory control
QSPCENV	CHAR(10)	Special environment
≫ QSPLFACN	CHAR(10)	Spooled file action≪
QSRLNBR	CHAR(8)	Serial number
QSRTSEQ	CHAR(20)	Sort sequence table
QSRVDMP	CHAR(10)	Service dump
QSTGLOWACN	CHAR(10)	Auxiliary storage lower limit action
QSTGLOWLMT	BINARY(4)	Auxiliary storage lower limit
QSTRPRTWTR	CHAR(1)	Start printer writer
QSTRUPPGM	CHAR(20)	Startup program name
QSTSMSG	CHAR(10)	Status messages
QSVRAUTITV	BINARY(4)	Server authentication interval
QSYSLIBL	ARRAY(15) of CHAR(10)	System library list
QTIME	CHAR(9)	System time
QTIMSEP	CHAR(1)	Time separator
QTOTJOB	BINARY(4)	Total jobs
QTSEPOOL	CHAR(10)	Time-slice end pool
QUPSDLYTIM	CHAR(20)	UPS delay time
QUPSMSGQ	CHAR(20)	UPS message queue
QUSEADPAUT	CHAR(10)	Use adopted authority

QUSRLIBL	ARRAY(25) of CHAR(10)	User library list
QUTCOFFSET	CHAR(5)	Coordinated universal time offset
QVFYOBJRST	CHAR(1)	Verify object on restore
QYEAR	CHAR(2)	Year

System Value Field Descriptions

Accounting level. QACGLVL is the accounting level. The possible values are:

NONE   No accounting information is written to a journal.

JOB   Job resource use is written to a journal.

PRINT The resources used for spooled and nonspooled print files are written to a journal.

Active jobs. QACTJOB is the initial number of active jobs for which auxiliary storage is to be allocated during IPL.

Additional active jobs. QADLACTJ specifies the additional number of active jobs for which auxiliary storage is to be allocated when the initial number of active jobs (the system value QACTJOB) is reached.

Additional storage. QADLSPLA specifies the additional storage to add to the spooling control block.

Additional total jobs. QADLTOTJ specifies the additional number of jobs for which auxiliary storage is to be allocated when the initial number of jobs (the system value QTOTJOB) is reached.

Allow object restore options. QALWOBJRST specifies a list of security options that are used when restoring objects to the system.

>> *ALL*   Allow all objects to be restored regardless of whether or not they have security-sensitive attributes or validation errors.

NONE   Does not allow objects with security-sensitve attributes to be restored.

ALWSYSSTT Allow programs, service programs, and modules with the system-state and inherit-state attribute to be restored.

ALWPGMADP Allow programs and service programs with the adopt attribute to be restored.

ALWPTF   Allow system-state and inherit-state programs, service programs, modules that adopt authority, objects that have the S_ISUID(set-user-ID) attribute enabled, and objects that have the S-ISGID(set-group-ID) attribute enabled to be restored to the system during PTF install.

ALWSETUID Allow restore of files that have the S_ISUID (set-user-ID) attribute enabled.

ALWSETGID Allow restore of files that have the S_ISGID (set-group-ID) attribute enabled.<<

ALWVLDERR Allow objects with validation errors to be restored.

Allow user domain. QALWUSRDMN is the allow user domain system value. It specifies a list of library names that can contain user domain objects.

ALL   All libraries and integrated file system directories on the system can contain user
               domain objects.

DIR   Any SOM object in a directory in the integrated file system can contain user domain
               objects. *DIR does not apply to the QSYS and QDLS file systems. *DIR is mutually
               exclusive with *ALL.

Library names A list of library names that can contain user domain objects.

Assistance level. QASTLVL is the assistance level system value. The value specifies the level of assistance available to users of the system.

BASIC   Operational Assistant level of system displays is available.

INTERMED   Intermediate level of system displays is available.

ADVANCED   Advanced level of system displays is available.

Attention program. QATNPGM is the attention program system value. The first 10 characters contain the program name and the last 10 characters contain the library name. The following special values are allowed:

ASSIST   The Operational Assistant main menu appears when the Attention key is pressed.

NONE   No attention program is called when the Attention key is pressed.

Auditing control. The QAUDCTL system value is the on/off switch for object- and user-level auditing. The values allowed are:

NONE   No auditing of objects and no auditing of user actions will be done on the system. In
               addition, no auditing that is controlled by the QAUDLVL system value will be done.

OBJAUD   Objects that have been selected by the Change Object Auditing (CHGOBJAUD) command
               will be audited.

AUDLVL   Changes controlled by the QAUDLVL system value and the AUDLVL parameter on the
               Change User Auditing (CHGUSRAUD) command will be audited.

Auditing end action. The QAUDENDACN system value indicates the action to be taken if auditing data cannot be written to the security auditing journal. These are the allowable values for the QAUDENDACN system value:

NOTIFY   The action that caused the audit to be attempted will continue after notification of
               failure to send the journal entry to the security auditing journal is sent to the
               QSYSOPR and QSYSMSG message queues.

PWRDWNSYS   The system ends with a system reference code (SRC) if sending of the audit data to the
               security audit journal fails. The system will then be brought up in a restricted state on
               the following IPL.

Auditing force level. The QAUDFRCLVL system value indicates to the system the number of auditing journal entries written to the security auditing journal before the auditing data is written to auxiliary storage. The following values are allowed:

0   The system will write the journal entries to auxiliary storage only when the system determines
               the journal entries should be written based on internal system processing.

1-100   The system will write the journal entries to auxiliary storage when this number of journal entries has been written to the security auditing journal.

Auditing level. QAUDLVL is the security auditing level. This system value specifies the level of security auditing that should occur on the system. The values allowed are:

*AUTFAIL   Authorization failures are audited.

*CREATE   The creation of objects is audited.

*DELETE   All object deletions are audited.

*JOBDTA   Actions by an audited user that affect a job will be audited.

*NONE   No auditing occurs on the system.

*OBJMGT   Function of generic objects is audited.

*OFCSRV   Auditing of OfficeVision licensed program.

*PGMADP   Program adoption.

*PGMFAIL   Integrity violations (for example, blocked instruction, validation value failure, and domain violation) are audited.

*PRTDTA   Printing of spool files or direct printing.

*SAVRST   Save and restore information is audited.

*SECURITY   All security-related functions are audited.

*SERVICE   Use of the system service tools by a user will be audited.

*SPLFDTA   Spool file auditing.

*SYSMGT   Use of system management functions by an audited user will be audited.

Automatic configuration for remote controllers. QAUTORMT allows the configuration of remote controllers. The possible values are:

0   Automatic configuration is off.

1   Automatic configuration is on.

Automatic system disabled reporting. The QAUTOSPRPT system value controls the automatic problem reporting ability. The value allows the system to automatically report a problem. The possible values are:

0   Automatic system disabled reporting is off.

1   Automatic system disabled reporting is on.

Automatic IPL date and time. QIPLDATTIM is the system value for the date and time to automatically do an IPL of the system. It specifies a date and time when an automatic IPL should occur. The special value *NONE indicates that no timed automatic IPL is desired. The format of the field returned is CYYMMDDHHMMSS, where C is the century, YY is the year, MM is the month, DD is the day, HH is the hour, MM is the minute, and SS is the second. A 0 for the century flag indicates years 19*xx*, and a 1 indicates years 20*xx*.

Automatic configuration for virtual device. QAUTOVRT is the system value for automatic configuration of virtual devices. This is the number of virtual devices that the user wants to have automatically configured. The possible values are:

0-32500 The number of virtual devices that the user wants to have automatically configured.

32767 There is no maximum number of virtual devices that the user wants to have automatically configured.

-1 The program registered for the Virtual Device Selection (QIBM_QPA_DEVSEL) exit point is called when a virtual device description needs to be selected or automatically created by the system. If the program registered for the exit point does not exist or if it returns with an error, the system will handle the situation as if the QAUTOVRT system value is set to 0.

Automatic configuration indicator. The QAUTOCFG system value automatically configures devices. The value specifies whether devices that are added to the system are configured automatically.

0 Automatic configuration is off.

1 Automatic configuration is on.

Auxiliary storage lower limit. QSTGLOWLMT is the percentage (in 10 thousandths) of the system auxiliary storage pool that remains available when the critical storage lower limit is reached. For example, a value of 50000 in binary would be 5.0000.

Auxiliary storage lower limit action. QSTGLOWACN is the action taken when the auxiliary storage lower limit (QSTGLOWLMT system value) is reached. The possible actions are:

*MSG Message CPI099C is sent to the QSYSMSG and the QSYSOPR message queues. (This message is also sent for each of the following actions.)

*CRITMSG Message CPI099B is sent to the user who is specified by the Critical messages to user service attribute. Service attributes can be changed by using the Change Service Attributes (CHGSRVA) command.

*REGFAC A job is submitted to run any exit programs that are registered for the QIBM_QWC_QSTGLOWACN exit point.

*ENDSYS The system is ended and left in the restricted state.

*PWRDWNSYS The system is powered down immediately and restarted.

Base activity level. QBASACTLVL is the base-storage-pool activity level. This value indicates how many system and user jobs can compete at the same time for storage in the base storage pool.

Base pool minimum size. QBASPOOL is the minimum size of the base storage pool. The base pool contains all main storage not allocated by other pools. QBASPOOL is specified in kilobytes.

Book and bookshelf search path. QBOOKPATH specifies which directories should be searched for books.

Century indicator. QCENTURY specifies the century value for the system date. The possible values are:

0 Indicates years 19*xx*.

1 Indicates years 20*xx*.

Character identifier control. QCHRIDCTL specifies the character identifier control for the job. This attribute controls the type of CCSID conversion that occurs for display files, printer files, and panel groups. The *CHRIDCTL special value must be specified on the CHRID command parameter on the create, change, or override command for display files, printer files, and panel groups before this attribute will be used. The possible values are:

DEVD   The *DEVD special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

JOBCCSID   The *JOBCCSID special value performs the same function as on the CHRID command parameter for display files, printer files, and panel groups.

Character set and code page. QCHRID is the default character set and code page. The QCHRID system value is retrieved as a single character value; the first 10 characters contain the character set identifier right-justified. For example, the value 101 would be retrieved as 0000000101. The last 10 characters contain the code page identifier right-justified. For example, the value 37 would be retrieved as 0000000037.

Coded character set identifier. QCCSID is the system value for coded character set identifiers.

Communication arbiters. QCMNARB specifies the number of communication arbiter jobs. The possible values are:

CALC   The operating system calculates the number of communication arbiter jobs.

0-99   The number of communication arbiter jobs.

Communications recovery limit. QCMNRCYLMT is the system value for communications recovery limits. The QCMNRCYLMT system value is retrieved as a 20-character value; the first 10 characters contain the count limit right-justified. For example, the value 7 would be retrieved as 0000000007. The last 10 characters contain the time interval right-justified. For example, the value 117 would be retrieved as 0000000117.

Configuration message queue. QCFGMSGQ is the configuration message queue system value. This message queue can be used to receive messages associated with configuration objects, such as lines and controllers. The first 10 characters contain the message queue name and the last 10 characters contain the library name.

Console name. QCONSOLE is the console name. This value specifies the name of the display device that is the console.

Controlling subsystem. QCTLSBSD is the controlling subsystem description. The controlling subsystem is the first subsystem to start after an IPL. The value of QCTLSBSD is a 20-character list of up to two 10-character values in which the first is the subsystem description name and the second is the library name.

Coordinated universal time offset. QUTCOFFSET is the system value indicating the difference in hours and minutes between Universal Time Coordinated (UTC), also known as Greenwich mean time, and the current system (local) time.

Country or region identifier. QCNTRYID is the system value for the country or region identifier. This value specifies the country or region identifier to be used as the default on the system.

Create authority. QCRTAUT is the create authority system value. This value allows the default public authority for the create (CRTxxx) commands to be set system-wide. The values allowed are:

CHANGE   Allows you to change the contents of an object.

ALL   Allows you to read, change, delete, and manage the security of an object.

USE   Allows you to create an object, to display the contents of an object, or to refer to the contents of an attached object when a command being requested must access attached objects and their contents.

EXCLUDE   Allows no access to an object.

Create object auditing. The QCRTOBJAUD system value indicates the default auditing value for new objects created into a library on the system. These are the allowable values for the QCRTOBJAUD system value.

NONE   No auditing entries are sent for this object when it is used or changed.

USRPRF   Auditing entries are sent for this object when it is used or changed by a user who is currently being audited. If the user who uses or changes this object is not being audited, no auditing entries are sent. To audit a user, you must use the Change User Auditing (CHGUSRAUD) command to change the user profile to that user profile.

CHANGE   Auditing entries are sent for this object when it is changed.

ALL   Auditing entries are sent for this object when it is used or changed.

Currency symbol. QCURSYM is the system value for the currency symbol. QCURSYM can be any character except blank, hyphen (-), ampersand (&), asterisk (*), or zero (0).

»Database file statistics collection. QDBFSTCCOL is the system value that specifies the type of statistic collection requests that will be allowed to be processed in the background by the database statistics system job, QDBFSTCCOL. Statistic collections which are requested by either a user or automatically by the database manager to be processed in the foreground are not affected by this system value. The values for QDBFSTCCOL can be:

NONE   No database file statistics collections are allowed to be processed by the database statistics system job.

USER   Only user requested database file statistics collections are allowed to be processed by the database statistics system job.

SYSTEM   Only automatically generated statistic collections requested by the database manager are allowed to be processed by the database statistics system job.

ALL   All user requested database statistics collections and statistic collections automatically requested by the database manager are allowed to be processed by the database statistics system job.«

Database recovery wait. QDBRCVYWT is the database recovery wait indicator. QDBRCVYWT can be:

0   Does not wait for database recovery to complete before completing the IPL.

1   Waits for database recovery to complete before completing the IPL.

Date format. QDATFMT is the system date format. This system value can be YMD, MDY, DMY, or JUL (Julian format), where Y equals year, M equals month, and D equals day.

Date separator. QDATSEP is the character separator for dates. QDATSEP can be slash (/), hyphen (-), period (.), comma (,), or blank.

Day. QDAY is the system value for the day of the month or year (if the date format is Julian). For Julian dates only, QDAY is a 3-character value (001 through 366).

Day of the week. QDAYOFWEEK specifies the day of the week. This value may not be set correctly if your system is not using the Gregorian calendar. The possible values are:

SUN   Sunday

MON   Monday

TUE   Tuesday

WED   Wednesday

THU   Thursday

FRI   Friday

SAT   Saturday

Days password valid. QPWDEXPITV is the system value for the password expiration interval. It controls the number of days that passwords are valid by keeping track of the number of days since you changed your password or created a user profile. The possible values are:

NOMAX   A password can be used an unlimited number of days.

1-366   The number of days before the password cannot be used.

DBCS installed. QIGC is the DBCS version indicator. This value specifies if the DBCS version of the system is installed. QIGC can be:

0   A DBCS version is not installed.

1   A DBCS version is installed.

Decimal format. QDECFMT is the decimal format. QDECFMT must be one of the following characters:

blank   Uses a period for a decimal point, a comma for a 3-digit grouping character, and zero-suppress to the left of the decimal point.

J   Uses a comma for a decimal point and a period for a 3-digit grouping character. The zero-suppression character is in the second position (rather than the first) to the left of the decimal notation. Balances with zero values to the left of the comma are written with one leading zero (0,04). The J entry also overrides any edit codes that might suppress the leading zero.

I   Uses a comma for a decimal point, a period for a 3-digit grouping character, and zero-suppress to the left of the decimal point.

Device naming convention. QDEVNAMING is the device naming convention. This value specifies what naming convention is used when the system automatically creates device descriptions. QDEVNAMING must be one of the following values:

NORMAL   Naming conventions should follow iSeries standards.

S36   Naming conventions should follow System/36 standards.

DEVADR   Device names are derived from the device address.

Device recovery action. QDEVRCYACN specifies what action to take when an I/O error occurs for an interactive job's work station. The values for QDEVRCYACN are:

MSG   Signals the I/O error message to the user's application program.

DSCENDRQS   Disconnects the job. When signing on again, a cancel request function is performed to return control of the job back to the last request level.

DSCMSG   Disconnects the job. When signing on again, an error message is sent to the user's application.

ENDJOB   Ends the job. A job log is produced for the job.

ENDJOBNOLIST   Ends the job. A job log is not produced for the job.

Disconnect job interval. QDSCJOBITV indicates the length of time, in minutes, an interactive job can be disconnected before it is ended. The values for QDSCJOBITV are:

5-1440   The range of the disconnect interval.

NONE   There is no disconnect interval.

Double-byte coded font name. QIGCCDEFNT is the system value for the double-byte coded font name. QIGCCDEFNT is a 20-character list of up to two values in which the first 10 characters contain the coded font name and the last 10 characters contain the library name. *NONE means no coded font is identified to the system.

Double-byte coded font point size. QIGCFNTSIZ is the system value for the double-byte coded font point size. The values for QIGCFNTSIZ are:

0   There is no defined double-byte coded font point size.

1-9999   The double-byte coded font point size in tenths. For example, a value of 9999 in binary would be 999.9.

Duplicate password. QPWDRQDDIF controls duplicate passwords. The possible values are:

0   A password can be the same as any previously used password (except the immediately preceding password).

1   A password must be different from the previous 32 passwords.

2   A password must be different from the previous 24 passwords.

3   A password must be different from the previous 18 passwords.

4   A password must be different from the previous 12 passwords.

5   A password must be different from the previous 10 passwords.

6   A password must be different from the previous 8 passwords.

7   A password must be different from the previous 6 passwords.

8   A password must be different from the previous 4 passwords.

Dynamic priority adjustment. The QDYNPTYADJ system value controls the dynamic priority

adjustment. Possible values are as follows:

0   Dynamic priority adjustment is off.

1   Dynamic priority adjustment is on.

Dynamic priority scheduler. The QDYNPTYSCD system value controls the dynamic priority scheduler algorithm. The value allows the use of the dynamic priority scheduler. Possible values are as follows:

0   Dynamic priority scheduler is off.

1   Dynamic priority scheduler is on.

≫**Force conversion on restore.** QFRCCVNRST is the system value that allows you to specify whether or not to convert the following object types during a restore: program (*PGM), service program (*SRVPGM), SQL package (*SQLPKG), and module (*MODULE). The possible values for QFRCCVNRST are as follows:

0   Do not convert anything.

1   Objects with validation errors will be converted.

2   Objects requiring conversion to be used on the current version of the operating system and objects with validation errors will be converted.

3   Objects suspected of having been tampered with, objects containing validation errors, and objects requiring conversion to be used by the current version of the operating system will be converted.

4   Objects that contain sufficient creation data to be converted and do not have valid digital signatures will be converted.

5   Objects that contain sufficient creation data will be converted.

6   All objects that do not have valid digital signatures will be converted.

7   All objects will be converted.

Any object that should be converted but cannot be converted will not be restored.≪

History log size. QHSTLOGSIZ is the maximum number of records for each version of the history log.

Hour. QHOUR is the system value for the hour of the day. Hours are based on a 24-hour clock. Its value can range from 00 through 23.

Inactive job time-out. QINACTITV specifies the inactive job time-out interval in minutes. It specifies when the system takes action on inactive interactive jobs. QINACTITV must be one of the following values:

*NONE   The system does not check for inactive interactive jobs.

5-300   The number of minutes a job can be inactive before action is taken.

Inactive message queue. QINACTMSGQ is the system value for the inactive message queue. QINACTMSGQ is a 20-character list of up to two 10-character values where the first is the message queue name and the second is the library name. The following special values are allowed.

*DSCJOB   The interactive job is disconnected, as is any secondary or group job associated with it.

ENDJOB   The interactive job is ended, along with any secondary job and any group job associated with it.

Initial spooling size. QJOBSPLA specifies the initial size of the spooling control block for a job.

IPL action with console problem. QSCPFCONS is the IPL action with a console problem indicator. This value specifies whether the IPL is to continue unattended or ends when the console is not operational when performing an attended IPL. QSCPFCONS can be:

0   End system.

1   Continue the IPL unattended.

IPL status. QIPLSTS is the IPL status indicator. This value indicates what form of IPL has occurred.

0   Operator panel IPL.

1   Automatic IPL after power restored.

2   Restart IPL.

3   Time-of-day IPL.

4   Remote IPL.

IPL type. QIPLTYPE indicates the type of IPL to perform. This value specifies the type of IPL performed when the system is powered on manually with the key in the normal position. QIPLTYPE can be:

0   Unattended.

1   Attended with dedicated service tools.

2   Attended with console in debug mode.

Job message queue full. QJOBMSGQFL specifies if the job message queue should be allowed to wrap.

NOWRAP   When the job message queue is full, do not wrap. This action causes the job to end.

WRAP   When the job message queue is full, wrap to the beginning and start filling again.

PRTWRAP   When the job message queue is full, wrap the message queue and print the messages that are being overlaid because of the wrapping.

Job message queue initial size. QJOBMSGQSZ specifies the initial size of the job message queue. QJOBMSGQSZ is numeric and is specified in kilobytes.

Job message queue maximum size. QJOBMSGQMX specifies the maximum size of the job message queue. QJOBMSGQMX is numeric and is specified in megabytes.

Keyboard buffer. QKBDBUF specifies whether the type-ahead feature and Attention key buffering option should be used.

TYPEAHEAD   The type-ahead feature is turned on, and the Attention key buffering option is turned off.

NO   The type-ahead feature and the Attention key buffering option are turned off.

YES	The type-ahead feature and the Attention key buffering option are turned on.

Keyboard type. QKBDTYPE specifies the language character set for the keyboard.

Language identifier. QLANGID is the system value for the language identifier. This system value specifies the language identifier to be used as the default for the system.

Leap year adjustment. QLEAPADJ is the system value for leap year adjustment. It is used to adjust the system calendar algorithm for the leap year in different calendar systems.

Library locking level. The QLIBLCKLVL system value controls whether libraries in a job's library search list are locked by that job. The *SHRRD locks prevent other jobs from deleting or renaming the libraries. System jobs, subsystem monitor jobs, and secondary threads do not lock libraries in their library search list. A change to this system value takes effect for all jobs that become active after the change. The shipped value is 1. The possible values are as follows:

0   Libraries in a user job's library search list are not locked.

1   Libraries in a user job's library search list are locked by that job.

Limit adjacent digits. QPWDLMTAJC limits adjacent digits in a password. It specifies whether adjacent digits are allowed in passwords. The possible values are:

0   Adjacent digits are allowed in passwords.

1   Adjacent digits are not allowed in passwords.

Limit character positions. Limit password character positions. This system value controls the position of characters in a new password. This prevents the user from specifying the same character in a password corresponding to the same position in the previous password.

A change to this system value takes effect the next time a password is changed. The shipped value is 0.

0   The same characters can be used in a position corresponding to the same position in the previous password.

1   The same character cannot be used in a position corresponding to the same position in the previous password.

Limit characters. QPWDLMTCHR limits the use of certain characters in a password. The possible values are:

NONE	There are no restricted characters.
restricted-characters	Up to 10 restricted characters can be specified. Valid characters are A through Z, 0 through 9, and special characters such as number sign (#), dollar ($), underscore (--), or at sign (@).

Note: This system value is ignored if the system is operating at QPWDLVL (password level) 2 or 3.

Limit device session. QLMTDEVSSN is the system value for limiting device sessions. It controls whether a user can sign-on at more than one work station.

0   A user can sign-on at more than one device.

1   A user cannot sign-on at more than one device.

Limit repeat characters. QPWDLMTREP limits the use of repeating characters in a password. The possible values are:

0   Characters can be used more than once.

1   Characters cannot be used more than once.

2   Characters can be used more than once but cannot be repeated consecutively.

Limit security officer. QLMTSECOFR is the system value for limiting QSECOFR device access. It controls whether users with *ALLOBJ or *SERVICE special authority need explicit authority to specific work stations. The possible values are:

0   A user with *ALLOBJ or *SERVICE special authority can sign-on any device.

1   A user with *ALLOBJ or *SERVICE special authority can sign-on only at a device to which they have explicit authority.

Locale path name. The QLOCALE system value specifies the locale object that is to be used. The possible values include a valid path name or one of the following special values:

*NONE   No locale object is specified.

*C   A predefined locale object is to be used.

*POSIX   A predefined locale object is to be used.

The locale name is returned in UCS-2 in the following format:

BINARY(4)   CCSID of the returned locale path name

CHAR(2)   Country or region ID

CHAR(3)   Language ID

CHAR(3)   Reserved field

BINARY(4)   Flag byte

BINARY(4)   Number of bytes in the locale path name

CHAR(2)   Locale delimiter

CHAR(10)   Reserved field

CHAR(2048)   Locale path name

Note: If the locale name is either the special value *C or *POSIX, a length of 1 is returned. If *NONE is specified, a length of 0 is returned. These values are returned in the default CCSID of the job.

Machine pool size. QMCHPOOL is the size of the machine storage pool. The machine storage pool contains shared machine and OS/400 licensed programs. QMCHPOOL is specified in kilobytes.

Maximum activity level. QMAXACTLVL is the maximum activity level of the system. This is the number of jobs that can compete at the same time for main storage and processor resources.

Maximum number of jobs. QMAXJOB specifies the maximum number of jobs allowed on the system.

Maximum password length. QPWDMAXLEN specifies the maximum length of a password. It controls the maximum number of characters in a password. The possible values are:

1-128   The maximum number of characters that can be specified for a password. If the system is operating at QPWDLVL (password level) 0 or 1, the valid range is 1-10. If the system is operating at QPWDLVL 2 or 3, the valid range is 1-128.

Maximum job message queue initial size. QJOBMSGQTL is the maximum initial size of the job message queue. QJOBMSGQTL is numeric and is specified in kilobytes.

Maximum not valid sign-on. QMAXSIGN specifies the maximum number of incorrect sign-on attempts allowed. The possible values are:

1-25   The maximum number of sign-on attempts allowed.

*NOMAX   There is no maximum number of sign-on attempts.

Maximum sign-on action. QMAXSGNACN specifies the maximum sign-on attempts action or how the system reacts when the maximum number of consecutive incorrect sign-on attempts (the system value QMAXSIGN) is reached. The possible values are:

1   Varies off the device if limit is reached.

2   Disables the user profile if limit is reached.

3   Varies off the device and disables the user profile if the limit is reached.

Maximum spooled files per job. QMAXSPLF specifies the maximum number of spooled files that can be created per job. A job can have more than the maximum number of spooled files specified by this system value if the spooled files existed before the system value was set to a lower number.

Minimum password length. QPWDMINLEN specifies the minimum length of a password. It controls the minimum number of characters in a password. The possible values are:

1-128   The minimum number of characters that can be specified for a password. If the system is operating at QPWDLVL (password level) 0 or 1, the valid range is 1-10. If the system is operating at QPWDLVL 2 or 3, the valid range is 1-128.

Minimum problem retention. QPRBHLDITV allows you to specify the minimum number of days a problem is kept in the problem log. The range for this system value is 0 through 999 days.

Minute. QMINUTE is the system value for the minute of the hour. Its value can range from 00 through 59.

Month. QMONTH is the system value for the month of the year. It will be blank if the date format specified in system value QDATFMT is Julian (JUL). Its value can range from 1 through 12.

Multithreaded job action. QMLTTHDACN is the system value for multithreaded job action. This value controls the action to be taken when a function that may not be threadsafe is called in a multithreaded job. The possible values are:

1   Perform the function that is not threadsafe without sending a message.

2   Perform the function that is not threadsafe and send an informational message.

3   Do not perform the function that is not threadsafe.

Parallel processing degree. QQRYDEGREE specifies the parallel processing option, which will also determine the types of parallel processing allowed. There are two types of parallel processing: input/output (I/O) parallel processing and symmetric multiprocessing (SMP). With I/O parallel processing, the database manager can use multiple tasks for each query to do the I/O processing. The central processing unit (CPU) processing will still be done serially. With SMP the CPU and I/O processing is assigned to tasks that run the query in parallel. Actual CPU parallelism requires a system with multiple processors. SMP parallelism can only be used if the system feature DB2 Symmetric Multiprocessing for OS/400 is installed.

NONE   No parallel processing is allowed for database query processing.

IO   Any number of tasks may be used when the database query optimizer chooses to use I/O parallel processing for queries. SMP parallel processing is not allowed.

OPTIMIZE The query optimizer can choose to use any number of tasks for either I/O or SMP parallel processing to process the query. Use of parallel processing and the number of tasks used is determined with respect to the following:

   - The number of processors available in the system
   - This job's share of the amount of active memory available in the pool in which the job is run
   - Whether the expected elapsed time for the query is limited by CPU processing or I/O resources

MAX   The query optimizer can choose to use either I/O or SMP parallel processing to process the query. The choices made by the query optimizer will be similar to those made for the value *OPTIMIZE except the optimizer will assume that all active memory in the pool can be used to process the query.

Pass-through servers. QPASTHRSVR specifies the number of target display-station pass-through server jobs that are available to process iSeries display-station pass-through, iSeries Access work station function (WSF), and other 5250 emulation programs on programmable workstations. The possible values are:

CALC   The operating system calculates the number of server jobs.

0-100 The number of server jobs.

Password level. QPWDLVL specifies the level of password support on the system. The possible values are:

0   User profile passwords with a length of 1-10 characters are supported.

1   User profile passwords with a length of 1-10 characters are supported. iSeries NetServer passwords for Windows 95/98/ME clients will be removed from the system.

2   User profile passwords with a length of 1-128 characters are supported.

3   User profile passwords with a length of 1-128 characters are supported. iSeries NetServer passwords for Windows 95/98/ME clients will be removed from the system.

Note: If this system value has been changed since the last IPL, this value is not the password level the system is currently using. This value will be in effect after the next IPL.

Password validation program. QPWDVLDPGM provides the ability for a user-written program to do additional validation on passwords. The possible values are:

NONE	A validation program is not used.
REGFAC	The password validation program name will be retrieved from the registration facility.
program-specification	The first 10 characters contain the name of the validation program and the last 10 characters contain the library name where the validation program is located. This option can only be used if the system is operating at QPWDLVL (password level) 0 or 1.

Password validation program. QPWDVLDPGM provides the ability for a user-written program to do additional validation on passwords. The first 10 characters contain the name of the program and the last 10 characters contain the library name. *NONE means a validation program is not used.

Performance adjustment. QPFRADJ indicates whether the system should adjust values during IPL and dynamically for system pool sizes and activity levels.

0   No performance adjustment.

1   Performance adjustment at IPL.

2   Performance adjustment at IPL and dynamically.

3   Dynamic performance adjustment.

Position characters. QPWDPOSDIF controls the position of characters in a new password. This prevents the user from specifying the same character in a password corresponding to the same position in the previous password. The possible values are:

0   The same characters can be used in a position corresponding to the same position in the previous password.

1   The same characters cannot be used in a position corresponding to the same position in the previous password.

Power down limit. QPWRDWNLMT is the maximum amount of time an immediate power down can take before processing is ended (abnormal end).

Power restore IPL. QPWRRSTIPL specifies whether the system should automatically do an IPL when utility power is restored after a power failure. The possible values are:

0   Automatic IPL is not allowed.

1   Automatic IPL is allowed.

Previous end of system indicator. QABNORMSW is the previous end of system indicator. The possible values are:

0   Previous end of system was normal.

1   Previous end of system was abnormal.

Print key format. QPRTKEYFMT specifies whether border and header information is provided when the Print key is pressed. The possible values are:

NONE   The border and header information is not included with output from the Print key.

PRTBDR   The border information is included with output from the Print key.

PRTHDR   The header information is included with output from the Print key.

PRTALL   The border and header information is included with output from the Print key.

Print text. QPRTTXT is the print text. This system value is used to print up to 30 characters of text on the bottom of listings and separator pages.

Printer device. QPRTDEV is the default printer device description. This value specifies the default printer for the system.

Problem filter. QPRBFTR specifies the name of the filter object that the service activity manager uses when processing problems. QPRBFTR is a 20-character list of up to two 10-character values in which the first value is the problem filter name and the second is the library name. *NONE means no problem filter is in use.

Problem hold interval. QPRBHLDITV allows you to specify the minimum number of days a problem is kept in the problem log. After this time interval, the problem can be deleted using the Delete Problem (DLTPRB) command. The time interval starts as soon as it is put into the log.

Processor feature. QPRCFEAT is the processor feature. It is the processor feature-code level of the system.

Processor multitasking. The QPRCMLTTSK system value controls processor multitasking. Possible values are as follows:

0   Processor multitasking is off.

1   Processor multitasking is on.

Query processing time limit. QQRYTIMLMT specifies a limit that is compared to the estimated number of elapsed seconds that a query requires to run in order to determine if a database query is allowed to start.

NOMAX   There is no maximum number of estimated elapsed seconds.

0-2147352578 The number of seconds that is compared to the estimated number of elapsed seconds required to run a query. If the estimated elapsed seconds is greater than this value, the query is not started.

Reclaim spool storage. QRCLSPLSTG is reclaim spool storage system value. It allows for the automatic removal of empty spool database members. The values allowed are:

NOMAX   The maximum retention interval.

NONE   No retention interval.

1-366   Number of days empty spool database members are kept for new spooled file use.

Remote service attribute. The QRMTSRVATR system value controls the remote service problem analysis ability. The value allows the system to be analyzed from a remote system. The values for QRMTSRVATR are as follows:

0   Remote service attribute is off.

1   Remote service attribute is on.

Remote IPL. QRMTIPL is the remote power on and IPL indicator. This value specifies if remote power on and IPL can be started over a telephone line. The possible values are:

0   Remote power on and IPL are not allowed.

1   Remote power on and IPL are allowed.

Remote sign-on. QRMTSIGN specifies how the system handles remote sign-on requests. The user can specify a program and library to decide which remote sessions will be allowed and which user profiles can be automatically signed on from which locations. The first 10 characters contain the program name, and the last 10 characters contain the library name. QRMTSIGN can have the following values:

*FRCSIGNON   All remote sign-on sessions are required to go through normal sign-on processing.

*SAMEPRF   When the source and target user profile names are the same, the sign-on may be bypassed for remote sign-on attempts.

*VERIFY   After verifying that the user has access to the system, the system allows the user to bypass the sign-on.

*REJECT   No remote sign-on is allowed.

Required password digits. QPWDRQDDGT specifies whether a digit is required in a new password. The possible values are:

0   A numeric digit is not required in new passwords.

1   A numeric digit is required in new passwords.

Retain server security data. QRETSVRSEC specifies whether security-related information for IBM-provided client/server applications is retained. The possible values are:

0   Do not retain the security-related information.

1   Retain the security-related information.

Second. QSECOND is the system value for the second of the minute. Its value can range from 00 through 59.

Security level. QSECURITY is the system security level indicator. The possible values are:

10   The system does not require a password to sign-on. The user has access to all system resources.

20   The system requires a password to sign-on. The user has access to all system resources.

30   The system requires a password to sign-on, and users must have authority to access objects and system resources.

40   The system requires a password to sign-on, and users must have authority to access objects and system resources. Programs that try to access objects through interfaces that are not supported will fail.

50   The system requires a password to sign-on, and users must have authority to access objects and system resources. Security and integrity of the QTEMP library and user domain (*USRxxx) objects are enforced. (Use system value QALWUSRDMN to change which libraries allow *USRxxx objects.) Programs fail if they try to pass unsupported parameter values to supported interfaces or if they try to access objects through interfaces that are not supported.

Note: If this system value has been changed since the last IPL, this value is not the security level the system is currently using. This value will be in effect after the next IPL.

Serial number. QSRLNBR is the system serial number. An example of a serial number is 1001003.

Server authentication interval. QSVRAUTITV is the system value for the server authentication interval. The server authentication interval specifies the time interval of the server authentication in minutes. The following values are allowed:

1-108000   The authentication of the token expires at the end of the interval specified.

Service dump. QSRVDMP specifies whether service dumps for unmonitored escape messages are created. The values that are allowed are:

*DMPALLJOB   Service dumps will be created for all jobs.

*DMPSYSJOB   Service dumps will be created for only system jobs, not user jobs.

*DMPUSRJOB   Service dumps are created for only user jobs, not system jobs. System jobs include the system arbiter, subsystem monitors, LU services process, spool readers and writers, and the start-control-program-function (SCPF) job.

*NONE   Do not request dumps in any jobs.

Set job attributes from locale. The QSETJOBATR system value specifies the job attributes that are to be set from the job's locale. The possible values for QSETJOBATR are as follows:

*NONE   No attributes are set, or use any combination of the following:

*CCSID   Coded character set identifier

*DATFMT   Date format

*DATSEP   Date separator

*DECFMT   Decimal format

*SRTSEQ   Sort sequence

*TIMSEP   Time separator

Shared memory control. QSHRMEMCTL specifies whether or not users are allowed to use shared memory or mapped memory that has write capability. The allowed values are:

0   Users are not allowed to use shared memory or mapped memory that has write capability.

1   Users are allowed to use shared memory or mapped memory that has write capability.

Sign-on information. QDSPSGNINF is the system value for displaying sign-on information. The possible values are:

0   The sign-on information is not displayed.

1   The sign-on information is displayed.

Software error log. QSFWERRLOG specifies whether software errors should be logged by the system. The allowed values are:

*LOG   Software errors are logged.

*NOLOG   No logging occurs.

Sort sequence table. QSRTSEQ is the name of the table used for the sort sequence. The first 10 characters contain the name of the table, and the last 10 characters contain the library name. The values for QSRTSEQ are:

*HEX	No sort sequence table is used. The hexadecimal values of the characters are used to determine the sort sequence.
*LANGIDSHR	The sort sequence table used can contain the same weight for multiple characters. The shared weight sort table associated with the language specified in the LANGID parameter is used.
*LANGIDUNQ	The sort sequence table used must contain a unique weight for each character in the code page, and it is the unique weight sort table associated with the language specified in the LANGID parameter.
sort sequence table name	The name and library of the sort sequence table to be used.

Special environment. QSPCENV specifies the system environment used as the default for all users. The possible values are:

*NONE   You enter the iSeries environment when you sign-on.

*S36   You enter the System/36 environment when you sign-on.

≫**Spooled file action.** QSPLFACN specifies whether spooled files can be accessed through job interfaces once a job has completed its normal activity.

*KEEP   Spooled files are kept with the job when the job completes its activity.

*DETACH   Spooled files are detached from the job when the job completes its activity.≪

Start printer writer. QSTRPRTWTR specifies whether printer writers are started at IPL. QSTRPRTWTR can be:

0   Do not start printer writers.

1   Start printer writers.

Startup program name. QSTRUPPGM is the startup program. This value specifies the name of the program called from an autostart job when the controlling subsystem is started. The first 10 characters contain the program name, and the last 10 characters contain the library name. *NONE means the autostart job ends normally without calling a program.

Status messages. QSTSMSG specifies whether or not the status messages are displayed. The values

allowed are:

NORMAL Status messages are displayed.

NONE Status messages are not displayed.

System date. QDATE is the system date. QDATE is composed of the following system values: QCENTURY, QYEAR, QMONTH, and QDAY. The format of the field returned is CYYMMDD where C is the century, YY is the year, MM is the month, and DD is the day. A 0 for the century flag indicates years 19*xx*, and a 1 indicates years 20*xx*.

System library list. QSYSLIBL is the system part of the library list. The list can contain as many as 15 names.

System model. QMODEL is the system model number. It is the number or letters used to identify the model of the system.

System time. QTIME is the system value for the time of day. QTIME is composed of the following system values: QHOUR, QMINUTE, and QSECOND. QTIME has the format HHMMSSXXX, where HH equals hours, MM equals minutes, SS equals seconds, and XXX equals milliseconds.

Time separator. QTIMSEP is the character separator for time. QTIMSEP must be one of the following values: colon (:), period (.), comma (,), or blank.

Time-slice end pool. QTSEPOOL is the time-slice end pool. This value specifies whether interactive jobs should be moved to another main storage pool when they reach time-slice end. The values allowed are:

NONE Jobs are not moved to the base storage pool when time-slice end is reached.

BASE Jobs are moved to the base pool when time-slice end is reached.

Total jobs. QTOTJOB specifies the initial number of jobs for which auxiliary storage is allocated during IPL.

UPS delay time. The uninterruptible-power-supply (UPS) delay time specifies the amount of time that elapses before the system automatically powers down following a power failure. When a change in power activates the UPS, messages are sent to the UPS message queue (the system value QUPSMSGQ). This system value is meaningful only if your system has a battery power unit or has an uninterruptible power supply attached.

A change to this system value takes effect the next time there is a power failure. The shipped value is *CALC. The allowed values are:

BASIC Powers only the PRC, IOP cards, and Load Source direct-access storage device. The appropriate wait time, in seconds, is calculated. (This should be used only if you have the battery power unit or an uninterruptible power supply without every rack being connected.)

  Note: All other values indicate that all racks have an uninterruptible power supply.

CALC Calculates the appropriate wait time. In a secondary partition, the calculated wait time, rather than *CALC, is returned.

NOMAX Starts no action.

0 Automatically powers down the system.

1-99999 Powers down the system after the specified number of seconds.

The QUPSDLYTIM system value is in the form of a two-item list. The first item is the value the user specified on the CHGSYSVAL command. The second item is the delay time, which is either what the user specified, or, if *CALC or *BASIC is specified, the calculated delay time.

UPS message queue. The QUPSMSGQ system value is the message queue that is to receive uninterruptible-power-supply messages. QUPSMSGQ is a 20-character list of up to two values in which the first 10 characters contain the message queue name, and the last 10 characters contain the library name.

Use adopted authority. QUSEADPAUT specifies an authorization list that is used to control who can create, change, and update programs and service programs with the use adopted authority (USEADPAUT) attribute of *YES. The possible values are:

*NONE	All users can create, change, and update programs and service programs that use adopted authority.
authorization list name	The name of an authorization list that a user must have at least *USE authority to in order to create, change, and update programs and service programs that use adopted authority. Authority to the authorization list cannot come from adopted authority.

User library list. QUSRLIBL is the default for the user part of the library list. The list can contain as many as 25 names.

»**Verify object on restore.** QVFYOBJRST is the system value for verify object on restore. This value is used to specify the policy to be used for object signature verification during a restore operation. This value applies to objects of types: *CMD, *PGM, *SRVPGM, *SQLPKG and *MODULE. It also applies to *STMF objects which contain Java programs. The possible values are:

1	Do not verify signatures on restore. Restore all objects regardless of their signature.
2	Verify signatures on restore. Restore unsigned commands and user-state objects. Restore signed commands and user-state objects, even if the signatures are not valid. Restore inherit-state and system-state objects only if they have valid signatures.
3	Verify signatures on restore. Restore unsigned commands and user-state objects. Restore signed commands and user-state objects only if the signatures are valid. Restore inherit-state and system-state objects only if they have valid signatures.
4	Verify signatures on restore. Do not restore unsigned commands and user-state objects. Restore signed commands and user-state objects, even if the signatures are not valid. Restore inherit-state and system-state objects only if they have valid signatures.
5	Verify signatures on restore. Do not restore unsigned commands and user-state objects. Restore signed user-state objects only if the signatures are valid. Restore inherit-state and system-state objects only if they have valid signatures.«

Year. QYEAR is the system value that specifies the last 2 digits for the year. Its value can range from 0 through 99.

Error Messages

Message ID	Error Message Text
CPF1860 E	Value &1 in list not valid.

CPF1861 E	Length of the receiver variable not valid.
CPF1862 E	Number of values to retrieve not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

»Retrieve Thread Attribute (QWTRTVTA) API

```
Required Parameter Group:

    1       Receiver variable                       Output      Char(*)
    2       Length of receiver variable             Input       Binary(4)
    3       Format of receiver information          Input       Char(8)
    4       Job or thread identification information Input       Char(*)
    5       Format of job identification information Input       Char(8)
    6       Number of fields to return              Input       Binary(4)
    7       Key of fields to return                 Input       Array(*) of
                                                                Binary(4)
    8       Reset performance statistics            Input       Char(1)
    9       Error code                              I/O         Char(*)

Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes.
```

The Retrieve Thread Attribute (QWTRTVTA) API retrieves job and thread attributes that apply to the job or thread specified in the job or thread identification information parameter.

Authorities and Locks

The following authority restrictions apply only when the API is called for format name RTVT0200. All other format names have no authority restrictions.

Job Authority

When calling this API for format name RTVT0200, one of the following conditions must be met:

❍ The API must be called from within the job for which the information is being retrieved.

❍ The caller of the API must be running under a user profile that is the same as the job user identity of the job for which the information is being retrieved. The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the Work Management book on the V5R1 Supplemental Manuals Web site.

❍ The caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

❍ The caller of the API must be authorized to the Thread Control function of Operating System/400 through iSeries Navigator's Application Administration support. The Change Function Usage Information (QSYCHFUI) API, with a function ID of QIBM_SERVICE_THREAD, can be used to change the list of users that are allowed to retrieve information about a thread.

Required Parameter Group

Receiver variable

> OUTPUT; CHAR(*)

> The variable that is used to return the attribute information for the specified thread.

Length of receiver variable

> INPUT; BINARY(4)

> The length of the receiver variable provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format of receiver information

> INPUT; CHAR(8)

> The format of the information returned in the receiver variable. The possible format name is:

> | | |
> |---|---|
> | *RTVT0100* | See [Format RTVT0100](#) for details on the job or thread attribute information returned. |
> | *RTVT0200* | Library list information. See [Format RTVT0200](#) for details on the library information returned for the thread. |
> | *RTVT0300* | Elapsed performance statistics. See [Format RTVT0300](#) for details on the performance statistics returned for the specified thread. |

Job or thread identification information

> INPUT; CHAR(*)

> The information that is used to identify the job or thread within a job for which attribute information is to be returned. See [Format of job or thread identification information](#) for details.

Format of job or thread identification information

> INPUT; CHAR(8)

> The format of the job or thread identification information. The possible format names are:

> | | |
> |---|---|
> | *JIDF0100* | See [Format of job or thread identification information](#) for details on the job identification information. |
> | *JIDF0200* | See [Format of job or thread identification information](#) for details on the job identification information. |
> | | **Note:** If the thread handle is available, Format JIDF0200 provides a faster method of accessing a thread that is not the current thread than Format JIDF0100. |

Number of fields to return

> INPUT; BINARY(4)

The number of fields to return in the specified format.

Key of fields to be returned

INPUT; ARRAY(*) of BINARY(4)

The list of fields to be returned in the specified format. For a list of valid fields, see Valid Keys.

Reset status statistics

INPUT; CHAR(1)

The elapsed time and all the key fields that are based on the elapsed time are reset to zero. If a format other than RTVT0300 is specified, this field needs to be zero. The following special values may be specified:

0   The elapsed time and the key fields based on the elapsed time are not reset.

1   The elapsed time and the key fields based on the elapsed time are reset back to zero.

Error code

I/O; CHAR(*)

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

Format RTVT0100

The RTVT0100 format returns job or thread attribute information for the specified thread. For the list of keys that are valid for job attributes and thread attributes, see Keys for RTVT0100.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(2)	Reserved
36	24	BINARY(4), UNSIGNED	Returned thread handle
40	28	CHAR(8)	Returned thread identifier
48	30	CHAR(10)	Job status
58	3A	CHAR(2)	Reserved
60	3C	BINARY(4)	Offset to key fields
64	40	BINARY(4)	Number of fields returned
68	44	CHAR(*)	Reserved
These fields repeat, in the order listed, for		BINARY(4)	Length of field information returned
		BINARY(4)	Key field

the number of key field returned.	CHAR(1)	Type of data
	CHAR(3)	Reserved
	BINARY(4)	Length of data
	CHAR(*)	Data
	CHAR(*)	Reserved

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.

Data. The data returned for the key field.

Job name. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the work station where the job started; for a batch job, you specify the name in the command when you submit the job.

Job number. The system-assigned job number.

Job status. The status of the job. The valid values are:

ACTIVE   The job has started, and it can use system resources (processing unit, main storage, and so on). This does not guarantee that the job is currently running, however. For example, an active job may be in one of the following states where it is not in a position to use system resources:

- The Hold Job (HLDJOB) command holds the job; the Release the (RLSJOB) command allows the job to run again.
- The Transfer Group Job (TFRGRPJOB) or Transfer Secondary Job (TFRSECJOB) command suspends the job. When control returns to the job, the job can run again.
- The job is disconnected using the Disconnect Job (DSCJOB) command. When the interactive user signs back on, thereby connecting back into the job, the job can run again.
- The job is waiting for any reason. For example, when the job receives the reply for an inquiry message, the job can start running again.

JOBQ   The job is currently on a job queue. The job possibly was previously active and was placed back on the job queue because of the Transfer Job (TFRJOB) or Transfer Batch Job (TFRBCHJOB) command, or the job was never active because it was just submitted.

OUTQ   The job has completed running and has spooled output that has not yet printed.

Key field. The field returned. See Valid Keys for the list of valid keys.

Length of data. The length of the data returned for the field.

Length of field information returned. The total length of information returned for this field. This value is used to increment to the next field in the list.

Number of fields returned. The number of fields returned to the application.

Offset to key fields. The offset in characters (bytes) from the beginning of the receiver to the key fields array entry.

Reserved. An ignored field.

Returned thread handle. A value which addresses a particular thread within the job. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread. This field will be 0 when called to return attributes for a job.

Returned thread identifier. A value which uniquely identifies the thread within the job. This field will be 0 when called to return attributes for a job.

Type of data. The type of data returned.

  C   The data is returned in character format.

  B   The data is returned in binary format.

User name. The user profile under which the job is started. The user name is the same as the user profile name and can come from several different sources depending on the type of job.

RTVT0200 Format

The RTVT0200 format returns library list information for the specified thread. The special value of -1 for the thread indicator field (part of the JIDF0100 format) may not be used. For the list of keys that are valid for this format, see Keys for RTVT0200.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(2)	Reserved
36	24	BINARY(4), UNSIGNED	Returned thread handle
40	28	CHAR(8)	Returned thread identifier
48	30	BINARY(4)	Offset to libraries in system library list
52	34	BINARY(4)	Number of system libraries
56	38	BINARY(4)	Offset to libraries in product library list
60	3C	BINARY(4)	Number of product libraries
64	40	BINARY(4)	Offset to current library
68	44	BINARY(4)	Number of current libraries
72	48	BINARY(4)	Offset to libraries in user library list
76	4C	BINARY(4)	Number of user libraries
80	50	BINARY(4)	Length of one library array entry

84	54	BINARY(4)	Offset to ASP group information
88	58	BINARY(4)	Number of ASP group information entries
92	5C	BINARY(4)	Length of one ASP group information entry
See note		Array(*) of CHAR(*)	System library list (See Library array entry for format of library array entry.)
		Array(*) of CHAR(*)	Product libraries (See Library array entry for format of library array entry.)
		Array(*) of CHAR(*)	Current library (See Library array entry for format of library array entry.)
		Array(*) of CHAR(*)	User library list (See Library array entry for format of library array entry.)
		Array(*) of CHAR(*)	ASP group information entry (See ASP Group Information Entry for format of ASP group information entry.)

Note: The decimal and hexadecimal offsets depend on the number of libraries you have in the various parts of your library lists and on keys requested. The data is left-justified and padded with blanks on the right. The array is sequential. See the CL Programming 📖 book for the total number of libraries that can be returned to you.

Library array entry

The library array entry describes the data that is returned for each library entry in the array of libraries. The name of the library as well as some extended information about the library is returned with this format.

For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
The fields repeat for each library object returned in the array.		CHAR(10)	Library name
		CHAR(50)	Library text description
		BINARY(4)	Library ASP number
		CHAR(10)	Library ASP name.
		CHAR(*)	Reserved

ASP Group Information Entry

The ASP group information entry describes the data that is returned for each ASP group. The name of the ASP group is returned with this format. For details about the fields listed, see Field Descriptions.

Offset			
Dec	Hex	Type	Field

These fields repeat for each ASP group returned.	CHAR(10)	ASP group name
	CHAR(*)	Reserved

Field Descriptions

ASP group information. The list of Auxiliary Storage Pool (ASP) group information for the current thread. This information does not include the system ASP or the basic user ASPs.

ASP group name. The name of an ASP group being used by the thread. This is the name of the primary ASP device in an ASP group.

Bytes available. All of the available bytes for use in your application.The actual length depends on how many libraries are in the library list.

Bytes returned. The number of bytes returned to the user. This may be some but not all of the bytes available.

Current library. The name of the current library for the specified thread. If no current library exists, the number of current libraries field is zero and this field has no entry in the list.

Job name. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the work station where the job started; for a batch job, you specify the name in the command when you submit the job.

Job number. The system-assigned job number.

Length of one ASP group information entry. The length of an entry in the ASP group information. Zero indicates that ASP group information is not being returned.

Length of one library array entry. The length of an entry in one of the library list entries.

Library ASP name. The name of the ASP in which the library is located. The following special values may also be returned:

  SYSBAS   The library is located in the system ASP or a basic ASP.

  N   The name of the ASP cannot be determined.

Library ASP number. The numeric identifier of the ASP device containing the object's library. The following values may be returned:

  1   The library is located in the system ASP.

  2-32   The library is located in a basic ASP.

  33-255 The library is located in an independent ASP.

  -1   The ASP device cannot be determined.

Library name. The name of the library object.

Library text description. The text description of the library object. This field is blank if no text description is specified.

Number of ASP group information entries. The number of elements in entries in the ASP group information. Zero indicates that ASP group information is not being returned.

Number of current libraries. The number of libraries in the current part of the library list of the specified thread.

Number of product libraries. The number of product libraries found in the library list of the specified thread.

Number of system libraries. The number of libraries in the system part of the thread's library list. This value will be zero if system libraries were not requested.

Number of user libraries. The number of libraries in the thread's user library list. This value will be zero if user libraries were not requested.

Offset to ASP group information. The offset in characters (bytes) from the beginning of the receiver to the first ASP group information entry. Zero indicates that ASP group information is not being returned.

Offset to current library. The offset in characters (bytes) from the beginning of the receiver to the current library array entry.

Offset to libraries in product library list. The offset in characters (bytes) from the beginning of the receiver to the first product library array entry.

Offset to libraries in system library list. The offset in characters (bytes) from the beginning of the receiver to the first system library array entry.

Offset to libraries in user library list. The offset in characters (bytes) from the beginning of the receiver to the first user library array entry.

Product libraries. The libraries that contain product information for the specified thread. If no product libraries exist, the number of product libraries field is zero and this field has no entry in the list.

Reserved. An ignored field.

Returned thread handle. A value which addresses a particular thread within the job. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread.

Returned thread identifier. A value which uniquely identifies the thread within the job.

System library list. The system portion of the library list of the specified thread.

User library list. The user portion of the library list for the specified thread.

User name. The user profile under which the job is started. The user name is the same as the user profile name and can come from several different sources depending on the type of job.

Format RTVT0300

The RTVT0300 format returns performance statistics information, calculated over an elapsed time, for the specified thread. The special value of -1 for the thread indicator field (part of the JIDF0100 format) may not be used. For the list of keys that are valid for this format, see Keys for RTVT0300.

Offset		

Dec	Hex	Type	Field
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User name
28	1C	CHAR(6)	Job number
34	22	CHAR(2)	Reserved
36	24	BINARY(4), UNSIGNED	Returned thread handle
40	28	CHAR(8)	Returned thread identifier
48	30	BINARY(8), UNSIGNED	Elapsed time
56	38	BINARY(4)	Offset to key fields
60	3C	BINARY(4)	Number of fields returned
64	40	CHAR(*)	Reserved
These fields repeat, in the order listed, for the number of key field returned.		BINARY(4)	Length of field information returned
		BINARY(4)	Key field
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data
		CHAR(*)	Data
		CHAR(*)	Reserved

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.

Data. The data returned for the key field.

Elapsed time. The time, in milliseconds, that has elapsed between the measurement start time and the current system time. This value is 0 the first time this API is called by this job. The measurement start is set the first time this API is called and when the reset status statistics is set to reset the elapsed time.

Job name. The name of the job as identified to the system. For an interactive job, the system assigns the job the name of the work station where the job started; for a batch job, you specify the name in the command when you submit the job.

Job number. The system-assigned job number.

Key field. The field returned. See Valid Keys for the list of valid keys.

Length of data. The length of the data returned for the field.

Length of field information returned. The total length of information returned for this field. This value is

used to increment to the next field in the list.

Number of fields returned. The number of fields returned to the application.

Reserved. An ignored field.

Returned thread handle. A value which addresses a particular thread within the job. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread.

Returned thread identifier. A value which uniquely identifies the thread within the job.

Type of data. The type of data returned.

   C   The data is returned in character format.

   B   The data is returned in binary format.

User name. The user profile under which the job is started. The user name is the same as the user profile name and can come from several different sources depending on the type of job.

Format of job or thread identification information

Format JIDF0100 is the format of the information needed to identify the job or the thread for which the job or thread's attributes will be returned. This format is to be used for returning job or thread information. This format supports several special values that can help in identifying the thread.

Format JIDF0200 is the format of the information needed to identify the thread for which the thread's attributes will be returned. This format is to be used when referencing a specific thread that you already have the identification information for.

Note: If the thread handle is available, Format JIDF0200 provides a faster method of accessing a thread that is not the current thread than Format JIDF0100.

JIDF0100 Format

Offset		Type	Field
Dec	Hex	Type	Field
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

* The job in which this program is running. The job number and user name must contain blanks.

INT The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread identifier. A value that uniquely identifies a thread within a job. If a thread identifier is specified, a thread indicator must also be specified. If the thread indicator is not 0, this field must contain hexadecimal zeros.

Thread indicator. A value that is used to specify the thread within the job for which information is to be retrieved. If a thread indicator is specified, a thread identifier must also be specified. The following values are supported:

-1 Information should be retrieved for the job. The value for the fields requested will be retrieved from the job. If the value requested only resides in a thread, the value for the initial thread will be returned. For example, the Current user field only resides in the thread and the initial thread value will be returned. The returned thread identifier and the returned thread handle will be returned as hexadecimal zeros.

0 Information should be retrieved for the thread specified in the thread identifier field. If the value requested only resides in a job, the value for the job containing the thread will be returned.

1 Information should be retrieved for the thread that this program is currently running in. If the value requested only resides in a job, the value for the job containing the thread will be returned.

2 Information should be retrieved for the initial thread of the identified job. If the value requested only resides in a job, the value for the job containing the thread will be returned.

Note: For all of the supported values, the combination of the internal job identifier, job name, job number and user name fields must also identify the job containing the thread.

User name. A specific user profile name, or blanks when the job name specified is a special value.

JIDF0200 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier

42	2A	CHAR(2)	Reserved
44	2C	BINARY(4), UNSIGNED	Thread handle
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns 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.

Job name. A specific job name or one of the following special values:

   *   The job in which this program is running. The job number and user name must contain blanks.

  *INT   The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread handle. A value that addresses a particular thread within a job. While the thread identifier uniquely identifies the thread within the job, the thread handle can improve performance when referencing the thread. A valid thread handle must be specified. The thread handle is returned on several other interfaces.

Thread identifier. A value which uniquely identifies a thread within a job. A valid thread identifier must be specified.

User name. A specific user profile name, or blanks when the job name specified is a special value.

Valid Keys

The following tables indicate the valid keys for the formats specified.

Keys for RTVT0100

The following table contains a list of the valid keys for format RTVT0100. See Key Field Descriptions for the descriptions of the valid key attributes. This table contains the keys that are valid for job attributes and thread attributes.

The Scope column defines the location of the attribute. The attribute is either scoped to the job or to the thread. If a job scoped attribute is requested for a thread, the value from the job containing the thread will be returned. If a thread scoped attribute is requested for a job, the value from the initial thread will be returned.

Key	Type	Description	Scope
0101	CHAR(4)	Active job status	Job

0102	CHAR(1)	Allow multiple threads	Job
0103	CHAR(4)	Active job status for jobs ending	Job
0104	CHAR(*)	ASP group information	Thread
0201	CHAR(10)	Break message handling	Job
0302	BINARY(4)	Coded character set ID	Job
0303	CHAR(2)	Country or region ID	Job
0305	CHAR(10)	Current user profile	Thread
0311	CHAR(10)	Character identifier control	Job
0312	BINARY(8, UNSIGNED)	Processing unit time used - total for the job	Job
0313	BINARY(8, UNSIGNED)	Processing unit time used for database - total for the job	Job
0319	BINARY(8, UNSIGNED)	Processing unit time used - total for the thread	Thread
0320	BINARY(8, UNSIGNED)	Processing unit time used for database - total for the thread	Thread
0326	CHAR(45)	Client IP address - IPv4 or IPv6	Thread
0401	CHAR(13)	Date and time job became active	Job
0402	CHAR(13)	Date and time job entered system	Job
0403	CHAR(8)	Date and time job is scheduled to run	Job
0404	CHAR(8)	Date and time job was put on this job queue	Job
0405	CHAR(4)	Date format	Job
0406	CHAR(1)	Date separator	Job
0407	CHAR(1)	DBCS-capable	Job
0408	CHAR(10)	DDM conversation handling	Job
0409	BINARY(4)	Default wait	Job
0410	CHAR(13)	Device recovery action	Job
0412	BINARY(4)	Default coded character set identifier	Job
0413	CHAR(1)	Decimal format	Job
0415	BINARY(8, UNSIGNED)	Disk I/O count - total for the job	Job
0418	CHAR(13)	Date and time job ended	Job
0420	BINARY(8, UNSIGNED)	Disk I/O count - total for the thread	Thread
0501	BINARY(4)	End severity	Job
0502	CHAR(1)	End status	Job
0504	CHAR(10)	Extended object attribute of entity thread is waiting on	Thread
0601	CHAR(10)	Function name	Job
0602	CHAR(1)	Function type	Job
0702	CHAR(10)	Group profile name	Thread
0703	CHAR(150)	Group profile name - supplemental	Thread
0901	CHAR(10)	Inquiry message reply	Job
1001	CHAR(15)	Job accounting code	Job
1002	CHAR(7)	Job date	Job
1003	CHAR(20)	Job description name - qualified	Job
1004	CHAR(20)	Job queue name - qualified	Job

1005	CHAR(2)	Job queue priority	Job
1006	CHAR(8)	Job switches	Job
1007	CHAR(10)	Job message queue full action	Job
1008	BINARY(4)	Job message queue maximum size	Job
1010	CHAR(1)	Job type	Job
1011	CHAR(1)	Job subtype	Job
1012	CHAR(10)	Job user identity	Job
1013	CHAR(1)	Job user identity setting	Job
1014	BINARY(4)	Job end reason	Job
1015	CHAR(1)	Job log pending	Job
1016	BINARY(4)	Job type - enhanced	Job
1201	CHAR(3)	Language ID	Job
1202	CHAR(1)	Logging level	Job
1203	CHAR(10)	Logging of CL programs	Job
1204	BINARY(4)	Logging severity	Job
1205	CHAR(10)	Logging text	Job
1206	CHAR(10)	Library of entity thread is waiting on	Thread
1302	BINARY(4)	Maximum processing unit time	Job
1304	BINARY(4)	Maximum threads	Job
1305	BINARY(4)	Maximum temporary storage in megabytes	Job
1306	CHAR(10)	Memory pool name	Job
1307	CHAR(1)	Message reply	Job
1407	CHAR(30)	Name of entity thread is waiting on	Thread
1501	CHAR(20)	Output queue name - qualified	Job
1502	CHAR(2)	Output queue priority	Job
1503	CHAR(10)	Object type of entity thread is waiting on	Thread
1601	CHAR(10)	Print key format	Job
1602	CHAR(30)	Print text	Job
1603	CHAR(10)	Printer device name	Job
1802	BINARY(4)	Run priority (job)	Job
1804	BINARY(4)	Run priority (thread)	Thread
1901	CHAR(20)	Sort sequence table - qualified	Job
1902	CHAR(10)	Status message handling	Job
1903	CHAR(10)	Status of job on the job queue	Job
1904	CHAR(26)	Submitter's job name - qualified	Job
1906	CHAR(20)	Subsystem description name - qualified	Job
1908	CHAR(10)	Special environment	Job
1911	CHAR(30)	Server type	Job
1982	CHAR(10)	Spooled file action	Job
2001	CHAR(1)	Time separator	Job
2002	BINARY(4)	Time slice	Job
2008	BINARY(4)	Thread count	Job

2009	BINARY(4)	Temporary storage used in megabytes	Job
2010	CHAR(4)	Thread status	Thread
2011	CHAR(1)	Thread type	Thread
2012	BINARY(4)	Thread hold count	Thread
2015	BINARY(4)	Type of entity thread is waiting on	Thread

Format of ASP Group Information

The ASP group information describes the data that is returned for key 104 from the RTVT0100 format.

Offset Dec	Offset Hex	Type	Field
0	0	BINARY(4)	Number of entries in ASP group information
4	4	BINARY(4)	Length of one ASP group information entry
This field repeats for each ASP group information entry.		CHAR(*)	ASP group information entry (See Format of ASP Group Information Entry for more information.)

Format of ASP Group Information Entry

The ASP group information entry describes the data that is returned for each ASP group. The name of the ASP group is returned with this format.

Offset Dec	Offset Hex	Type	Field
These fields repeat for each ASP group returned.		CHAR(10)	ASP group name
		CHAR(*)	Reserved

Keys for RTVT0200

The following table contains a list of the valid keys for format RTVT0200. See Key Field Descriptions for the descriptions of the valid key attributes.

All the library keys are scoped to the thread. See Library array entry for format of library array entry. These keys are valid only when retrieving the library list of a specific thread. The special value of -1 for the thread indicator field (part of the JIDF0100 format) may not be used. The job or thread identification format must specify a specific valid thread.

Key	Type	Description

0104	CHAR(*)	ASP group information (thread)
0310	CHAR(*)	Current library
1660	Array(*) of CHAR(*)	Product libraries
1980	Array(*) of CHAR(*)	System library list
2110	Array(*) of CHAR(*)	User library list
2703	Array(*) of CHAR(*)	All portions of the library list for format RTVT0200

Keys for RTVT0300

The following table contains a list of the valid keys for format RTVT0300. See Key Field Descriptions for the descriptions of the valid key attributes. All keys specified in this table are scoped to the thread. The special value of -1 for the thread indicator field (part of the JIDF0100 format) may not be used. The job or thread identification format must specify a specific valid thread.

Key	Type	Description
0321	BINARY(4)	Processing unit used - percent during the elapsed time (thread).
0322	BINARY(8), UNSIGNED	Processing unit used - time during the elapsed time (thread).
0323	BINARY(4)	Processing unit used for database - percent used during the elapsed time (thread).
0324	BINARY(8), UNSIGNED	Processing unit used for database - time during the elapsed time (thread) .
0419	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time (thread).
0421	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - asynchronous I/O (thread).
0422	BINARY(8), UNSIGNED	Disk I/O count during the elapsed time - synchronous I/O (thread).
1610	BINARY(8), UNSIGNED	Page fault count during the elapsed time (thread).

Key Field descriptions

The descriptions of all the valid key attributes are described in Work Management API Attribute Descriptions (WMAttrDesc).

Usage Notes

The conditions under which this API is threadsafe are as follows:

- Retrieving the attributes from a job - see [Retrieve Job Information](#) (QUSRJOBI) API for details on thread safety.

- Retrieving the attribute for a specific thread - this is thread safe.

Error Messages

Message ID	Error Message Text
CPF136A E	Job &3/&2/&1 not active.
CPF1866 E	Value &1 for number of fields to return not valid.
CPF1867 E	Value &1 in list not valid.
CPF18BF E	Thread &1 not found.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C20 E	Error found by program &1.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C3C E	Value for parameter &1 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.
CPF3C54 E	Job &3/&2/&1 currently not available.
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.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.

API introduced: V5R2

[Top](#) | [Work Management APIs](#) | [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.

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

Set Profile Exit Programs (QWTSETPX) API

Required Parameter Group:

1	Number of entries	Input	Binary(4)
2	Exit program flags	Input	Array(*) of Binary(4)
3	Format	Input	Char(8)
4	User ID	Input	Char(10)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: No

The Set Profile Exit Programs (QWTSETPX) API sets for the specified user ID the profile exit programs to call based on the format value. The value ATTN0100 sets the exit flags for attention key processing. The value SREQ0100 sets the flags for system request processing. For the specified user ID, each of the eight exit program flags may be set to the following:

0   No, do not call this exit program.

1   Yes, call this exit program.

-1   Same, do not change the value.

If all the values are set to 0 (No), no new actions are taken during the attention key processing or system request processing.

Each exit program flag that is set to 1 (Yes) by this API corresponds to the exit program number of the exit programs that are registered in the registration facility for the QIBM_QWT_PREATTNPGMS exit point or the QIBM_QWT_SYSREQPGMS exit point.

When attention key processing is activated by a job that is running under the specified user ID, only the exit programs that have a 1 (Yes) for that exit program flag are called.

When system request key processing is activated by a job that is running under the specified user ID, only the exit programs that have a 1 (Yes) for that exit program flag are called.

Authorities and Locks

User ID Authority
> *CHANGE

Required Parameter Group

Number of entries

> INPUT; BINARY(4)

> The number of exit program flags that are being passed in. The maximum number of flags is 8, and the minimum number of flags is 1.

Exit program flags

> INPUT; ARRAY(*) of BINARY(4)

> An array of a number of elements. The number of array elements must match the number of entries. The valid values for the array elements are as follows:

> *0*   No, do not call this exit program.

> *1*   Yes, call this exit program.

> *-1*   Do not change the value.

> The first element of the array corresponds to exit program number one for the exit point that is in the registration facility. The second array element corresponds to exit program number two and so on.

Format

> INPUT; CHAR(8)

> The format that is to be updated. Valid values are as follows:

> *ATTN0100*   The preattention program processing flags are to be updated.

> *SREQ0100*   The presystem request program processing flags are to be updated.

User ID

> INPUT; CHAR(10)

> The user ID name being updated. Valid values are as follows:

> **CURRENT*   The user ID of the job that is currently running is used.

> *User ID name*   The 10-character name that is entered is used.

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
CPF1666 E	Number of entries not in valid range.
CPF2204 E	User profile &1 not found.
CPF2213 E	Not able to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
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: V3R6

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

Auxiliary Storage Lower Limit Exit Program

> Required Parameter: None
>
> Exit Point Name: QIBM_QWC_QSTGLOWACN
>
> Exit Point Format Name: STGL0100

The Auxiliary Storage Lower Limit exit program is called when the available storage in the system auxiliary storage pool (ASP) goes below the lower limit. The exit program is called only if the QSTGLOWACN system value is set to *REGFAC. The lower limit is specified by the QSTGLOWLMT system value.

When the storage lower limit is reached, the operating system submits a job that calls the user-written exit program through the registration facility. There are no input or output parameters.

This exit point supports any number of exit programs. (For information about adding an exit program to an exit point, see the [Registration Facility APIs](#).)

If available storage in the system ASP is still below the auxiliary storage lower limit after the exit program has completed, another job is submitted to call the exit program after half an hour. (Changing the QSTGLOWACN and QSTGLOWLMT system values can cause the exit program to run again during this time.) Otherwise, the exit program will be called when available storage goes below the limit again.

The job is submitted to run in the QSYSWRK subsystem (using the QSYSJOBD job description) under the QPGMR user profile. No other action is taken if the job does not run.

»Notes:
1. The Auxiliary Storage Lower Limit exit point ignores any return codes or error messages that are sent from the exit programs.

2. It is recommended that the Auxiliary Storage Lower Limit exit program exist in a library in the system ASP or in a basic user ASP. It will not be found if it exists in a library in an independent ASP.«

Authorities and Locks

User Profile Authority
>   *ALLOBJ and *SECADM are needed to add exit programs to the registration facility.

Required Parameter

None.

Exit program introduced: V4R2

Job Notification Exit Point

Required Parameter: None

≫QSYSINC Member Name: EJOBNTFY≪

Exit Point Name: QIBM_QWT_JOBNOTIFY

Exit Point Format Name: NTFY0100

The Job Notification exit point can be used to log notification messages to data queues when OS/400 jobs go through the following transitions:

- A job is placed on a job queue.
- A job starts.
- A job ends.

The QIBM_QWT_JOBNOTIFY exit point registers a data queue and library, rather than an exit program and library. The program data that is associated with the data queue must contain the notification type, subsystem description, and subsystem description library.

The information will be retrieved from the registration facility when the subsystem starts, so the data queues must be registered before starting the subsystem. Any queues added to the registration facility after a subsystem is started will not be retrieved until the next time the subsystem starts.

While multiple subsystems can use the same data queue, each subsystem is limited to using a maximum of eight data queues. If more than eight data queues are registered for a subsystem, the specific data queues that will be selected are undefined.

If a job is submitted to a job queue that is not allocated by an active subsystem, a job queue notification message will be sent to a default data queue of QSYSDTAQ in library QSYS.

Use the Create Data Queue (CRTDTAQ) command to create any data queues to be used by this function, including the QSYSDTAQ mentioned previously. See Data Queue Attributes for additional information about the attributes of the data queues.

For the format of the job start and job end notification messages, see Format of Job Start and Job End Notification Messages. For the format of the job queue notification messages, see Format of Job Queue Notification Messages.

Authorities and Locks

None.

Required Parameter

None.

Program Data

When you register the data queue, the following is required for the program data.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(4)	Notification type
4	4	CHAR(10)	Subsystem description
14	E	CHAR(10)	Subsystem description library

Field Descriptions

Notification type. The type of notifications that are to be sent to the data queue. The following values are supported:

0001   Job start notifications are sent to this data queue.

0002   Job end notifications are sent to this data queue.

0003   Job start and job end notifications are sent to this data queue.

0004   Job queue notifications are sent to this data queue.

0005   Job start and job queue notifications are sent to this data queue.

0006   Job end and job queue notifications are sent to this data queue.

0007   Job start, job end, and job queue notifications are sent to this data queue.

Subsystem description. The name of the subsystem description for which this data queue is to be used. The following special value is supported:

*ANY   The data queue is used for all subsystems. When this value is specified, the subsystem description library is ignored.

Subsystem description library. The name of the library that contains the subsystem description. The following special value is supported:

*ANY   The data queue is used for any subsystems that match the subsystem description name, regardless of the library.

Data Queue Attributes

The following table lists several data queue attributes and the required values for the data queues that are used by this exit point.

Attribute	Value
Maximum entry length	74 or greater
Sequence	*KEYED
Key length	4

As shown in the previous table, the data queue entries are received by key. The following keys are used with these data queues:

0001   Data queue message is a job start notification message.

0002   Data queue message is a job end notification message.

0004   Data queue message is a job queue notification message.

Format of Job Start and Job End Notification Messages

For more information about this format, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Message identifier
10	A	CHAR(2)	Message format
12	C	CHAR(16)	Internal job identifier
28	1C	CHAR(26)	Qualified job name
54	36	CHAR(20)	Reserved

Format of Job Queue Notification Messages

For more information about this format, see Field Descriptions.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Message identifier
10	A	CHAR(2)	Message format
12	C	CHAR(16)	Internal job identifier
28	1C	CHAR(26)	Qualified job name
54	36	CHAR(20)	Qualified job queue name

Field Descriptions

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

Message format. The format of the data in the job notification message. This field is always set to 01 for job start and job end notification messages, and is always set to 02 for job queue notification messages.

Message identifier. The type of message that is represented by this data queue entry. This field is always set to *JOBNOTIFY.

Qualified job name. The name of the job that is associated with this notification message. The format of the qualified job name is a 10-character simple job name, a 10-character user name, and a 6-character job number. Each portion of the qualified job name is left-justified and padded with blanks on the right.

Qualified job queue name. The name of the job queue that the job was placed on, and the name of the library that contains the job queue. The format of the qualified name is a 10-character simple object name followed by a 10-character library name. Each portion of the qualified job queue name is left-justified and padded with blanks on the right.

Reserved. An ignored field.

Usage Notes

1. The notification messages may not be logged to the data queue if a DDM data queue is specified for this exit point.

2. ≫It is recommended that the data queue exist in a library in the system auxiliary storage pool (ASP) or in a basic user ASP. It will not be found if it exists in a library in an independent ASP.≪

Exit program introduced: V3R7

Power Down System Exit Program

Required Parameter: None

Exit Point Name: QIBM_QWC_PWRDWNSYS

Exit Point Format Name: PWRD0100

The Power Down System exit program is called when the Power Down System (PWRDWNSYS) or End System (ENDSYS) command is used.

When a user issues the PWRDWNSYS or ENDSYS command, the operating system calls the user-written exit program through the registration facility. The user-written exit program is not called if the system is already in restricted state, if the system is currently on auxiliary power, if the system is ending because the storage lower limit is reached with an action of *ENDSYS or *PWRDWNSYS, or if the system is a secondary partition and is powering down because the primary partition is powering down. There are no input or output parameters.

This exit point supports one exit program. (For information about adding an exit program to an exit point, see the [Registration Facility APIs](#).)

≫**Notes:**
1. The Power Down System exit point ignores any return codes or error messages that are sent from the exit program.

2. It is recommended that the Power Down System exit program exist in a library in the system auxiliary storage pool (ASP) or in a basic user ASP. It might not be found if it exists in a library in an independent ASP.≪

Authorities and Locks

User Profile Authority
> *ALLOBJ and *SECADM to add exit programs to the registration facility.

Required Parameter

None.

Exit program introduced: V4R2

Preattention Program Exit Program

Required Parameter: None

Exit Point Name: QIBM_QWT_PREATTNPGMS

Exit Point Format Name: ATTN0100

The Preattention Program exit program is called when the user presses the System Attention key.

When a user presses the System Attention key, the operating system calls the user-written exit program through the registration facility. There are no input or output parameters. After the exit programs from the registration facility are called, the system attention program is called.

This exit point supports eight exit programs. (For information about adding an exit program to an exit point, see the Registration Facility APIs.)

Before any of the exit programs in the registration facility are called, the user profile for each user intending to use this function will need to be updated. The Set Profile Exit Program (QWTSETPX) API will need to be called. This API will set for the requested user profile which of the eight allowed exit programs to call. The user can have none or all eight possible exit programs called. No exit programs will be called until the QWTSETPX API is called to set the values in the user profile. To display which exit programs will be called for a particular user, the Retrieve Profile Exit Program (QWTRTVPX) API can be called. (For more information about both of these APIs, see the Work Management APIs.)

Notes:
1. ≫The Preattention Program exit point ignores any return codes or error messages that are sent from the exit programs.

2. It is recommended that the Preattention Program exit program exist in a library in the system auxiliary storage pool (ASP) or in a basic user ASP. It might not be found if it exists in a library in an independent ASP. ≪

Authorities and Locks

User Profile Authority
   *ALLOBJ and *SECADM to add exit programs to the registration facility.

Required Parameter

None.

Exit program introduced: V3R7

Presystem Request Program Exit Program

```
Required Parameter:

  1     System Request menu display flag      I/O          Binary(4)

Optional Parameter:

  2     System Request user option data  I/O         CHAR(128)


Exit Point Name: QIBM_QWT_SYSREQPGMS

Exit Point Format Name: SREQ0100
```

The Presystem Request Program exit program is called when the user presses the System Request key.

When a user presses the System Request key, the operating system calls the user-written exit program through the registration facility. There are two parameters used for input and output. After the exit programs from the registration facility are called, the System Request menu is called based on the value that is returned in the System Request menu display flag. This is the first parameter. The second parameter contains the data the user entered on the System Request user option line. When the user presses the the System Request key an input line is displayed at the bottom of the screen. This allows the user to enter one of the option numbers and relevant parameter data for any of the options for the Systems Request menu. All data keyed into the input line, including option numbers, will be passed to the exit programs. This data will be passed and received from each of the exit programs called. This allows one program to modify the data and pass it to the next program called. This data will be passed to the user exit program in the CCSID of the job. The data that the last Presystem Request program returns will the data that is passed to the System Request menu. Normal error checking will be performed on the data at that time.

This exit point supports eight exit programs. (For information about adding an exit program to an exit point, see the Registration Facility APIs.)

Before any of the exit programs in the registration facility are called, the user profile for each user intending to use this function needs to be updated. The Set Profile Exit Program (QWTSETPX) API needs to be called. This API will set for the requested user profile which of the eight allowed exit programs to call. The user can have from zero to all eight possible exit programs called. No exit programs will be called until the QWTSETPX API is called to set the values in the user profile. To display which exit programs will be called for a particular user, the Retrieve Profile Exit Program (QWTRTVPX) API can be called. (For more information about both of these APIs, see the Work Management APIs.)

≫**Notes:**

1. It is recommended that a command line is not displayed from a presystem request exit program. There may be functions that do not work as expected when issued from this type of command line.

2. The Presystem Request exit point ignores any error messages that are sent from the exit programs. If multiple exit program are registered, all programs will be called regardless of errors that may occur.

3. The user-registered exit program will be called with only the first parameter unless the Pass user option data flag in the Program data section is turned on. If this flag is turned on, the user exit program will be called with both of the parameters.

4. It is recommended that the Presystem Request exit program exist in a library in the system auxiliary storage pool (ASP) or in a basic user ASP. It might not be found if it exists in a library in an independent ASP.«

Authorities and Locks

User Profile Authority
> *ALLOBJ and *SECADM to add exit programs to the registration facility.

Required Parameter Group

System Request menu display flag
> I/O; BINARY(4)

> Whether to display the System Request menu after the exit programs from the registration facility have been called. The default value of 1 means that the menu will be displayed. Any other value will mean not to display the menu. This value will be passed to all the exit programs. The value returned from the last program called is the value that will be used to determine if the menu should be displayed.

Optional Parameter Group

System Request user option data
> I/O; CHAR(128)

> The user-entered data from the System Request user option field. The default is all blanks.

Program Data

When you register the exit program, the following is required to enable passing of the System Request option data for the program data field.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(1)	Pass user option data flag

Field Descriptions

Pass user option data flag. This flag determines how many parameters need to be passed on the call to the user exit program. If the flag is set to a 1, then both of the parameters will be passed on the call to the exit program. If the flag is any other value, only the first parameter will be passed.

Exit program introduced: V3R7

Exit Program for Trace Job

```
Required Parameter Group:

 1    Trace record              Input      Char(*)


QSYSINC Member Name: EPDTRCJB
```

The Trace Job (TRCJOB) command prints trace records. If specified, the TRCJOB command calls a user-written program before printing the trace record. TRCJOB passes the trace record to the user-written program. If the user-written program changes the first two characters of the trace record to blanks or binary zeros, TRCJOB will not print the record.

See the online help for more information on the TRCJOB command.

Required Parameter

Trace record

> INPUT; CHAR(*)

> The format of the trace record depends on the trace record type.

Format of Trace Record

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Trace record type
1	1	CHAR(*)	Trace record data

Field Descriptions

Trace record type. This identifies the type of trace record:

X'00' Program flow

X'01' Data

X'02' Suspend

X'03' Resume

Trace record data. The format of the trace record data depends on the type of trace record.

Format of Program Flow Trace Record

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Program flow type
1	1	CHAR(8)	Time stamp
9	9	CHAR(10)	Program name
19	13	CHAR(10)	Program library name
29	1D	BINARY(2)	Call stack counter
31	1F	BINARY(2)	Number of deleted records
33	21	CHAR(2)	New instruction number
35	23	CHAR(2)	Old instruction number
37	25	CHAR(4)	CPU time used
41	29	BINARY(4)	Number of database pages read
45	2D	BINARY(4)	Number of nondatabase pages read
49	31	BINARY(4)	Number of pages written
53	35	BINARY(2)	Number of transfers to wait state
55	37	CHAR(10)	ILE module name
65	41	CHAR(10)	ILE module library name
75	4B	CHAR(256)	Procedure name
331	14B	BINARY(4)	Offset to long procedure name
335	14F	BINARY(4)	Length of long procedure name
*	*	CHAR(*)	Long procedure name

Field Descriptions

Call stack counter. The location in the call stack where the program received control.

CPU time used. The CPU time the program used.

ILE module library name. The name of the library containing the ILE module that received control.

ILE module name. The name of the ILE module that received control.

Length of long procedure name. The length of the name of the procedure that received control.

Long procedure name. The name of the procedure that received control. This field is filled regardless of the procedure name length.

New instruction number. The machine interface instruction number where the program received control.

Number of database pages read. The number of database pages that were read into main storage.

Number of deleted records. The number of successive records the exit program deleted.

Number of nondatabase pages read. The number of pages not associated with database that were read into main storage.

Number of pages written. The number of pages written from main storage to auxiliary storage.

Number of transfers to wait state. The number of transitions transferred to ineligible wait state.

Offset to long procedure name. The offset to the field that holds the procedure name

Old instruction number. The machine interface instruction number from which the program released control.

Procedure name. The name of the procedure that received control. This field is filled only if the procedure name is less than or equal to 256 bytes in length.

Program flow type. How the program received control:

X'01'   Another program called this program.

X'02'   Another program transferred control to this program.

X'03'   An event occurred, and this program is the event handler.

X'04'   An exception occurred, and this program is the exception handler.

X'05'   An exception occurred, and this is the internal or branch point exception handler.

X'06'   An internal exception handler returned to this program.

X'07'   This is a call to the exit program.

X'08'   This program called another program, and the other program returned control to this program.

X'09'   This program ended because it resignaled an exception.

X'0A'   An external exception handler returned to this program.

X'0B'   The end phase of the process is ended, ending this program.

X'0C'   An exception occurred for which no handler was specified.

X'0E'   A recursion level lower than this one was ended, causing this recursion level to end.

X'0F'   The call was canceled.

Program library name. The name of the library containing the program that received control.

Program name. The name of the program that received control.

Reserved. An ignored field.

Time stamp. The time when the event occurred.

Format of Data Trace Record

The data source field of the data trace record can be used to identify the correct format. There are four data trace record formats:

- Data management
- Message handler
- Command analyzer
- Other

Format of Data Management Data Trace Record

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Data source
1	1	CHAR(8)	Time stamp
9	9	CHAR(10)	Internal file name
19	13	CHAR(10)	Actual file name
29	1D	CHAR(10)	Actual file library name
39	27	CHAR(10)	Device description name
49	31	CHAR(10)	Program device name

Field Descriptions

Actual file library name. The name of the library the actual file is in. This will be *N if the file is an inline data file.

Actual file name. The name of the file that was used. This name will be different from the internal file name if the internal file was overridden.

Data source. Shows which function the data is from:

X'01' Open file

X'02' Close file

X'0B' Acquire device

X'0C' Release device

Device description name. For a device file, this is the device description name of the first device in the file. For a database file, this is the name of the member. For an inline data file, this is *N.

Internal file name. The name of the file in the program.

Program device name. The program device name of the device in the device file.

Time stamp. The time when the event occurred.

Format of Message Handler Data Trace Record

Offset			
Dec	Hex	Type	Field
0	0	CHAR(1)	Data source
1	1	CHAR(8)	Time stamp
9	9	CHAR(1)	Trace record type
10	A	CHAR(7)	Message identifier
17	11	CHAR(1)	Message type
18	12	BINARY(2)	Message severity
20	14	CHAR(10)	Message queue name
30	1E	BINARY(2)	Call stack number
32	20	CHAR(256)	Procedure name
288	120	CHAR(10)	ILE module name
298	12A	CHAR(10)	ILE module library name
308	134	CHAR(1)	Message queue type
309	135	CHAR(2)	Reserved
311	137	BINARY(4)	Offset to long procedure name
315	13B	BINARY(4)	Length of long procedure name
*	*	CHAR(*)	Long procedure name

Field Descriptions

Call stack number. The location in the call stack that the message handling function was called.

Data source. The function the data is from:

X'03'   Send message

X'04'   Reply to message

X'05'   Default exception handler

X'06'   Send status message

ILE module library name. The name of the library that contains the ILE module that sent or received the message.

ILE module name. The name of the ILE module that sent or received the message.

Length of procedure name. The length of the name of the procedure that sent or received the message.

Long procedure name. The name of the procedure that sent or received the message. This field is filled

regardless of the procedure name length.

Message identifier. The identifier of the message that was handled.

Message queue name. The name of the message queue the message was sent to or received from.

Message queue type. The type of message queue the message was sent to:

Q   The message was sent to a standard message queue.

blank   The message was sent to a nonstandard message queue.

Message severity. The severity of the message as defined in the message description.

Message type. The type of message that was handled:

X'01'   Completion

X'02'   Diagnostic

X'03'   Scope

X'04'   Informational

X'05'   Inquiry

X'06'   Sender's copy

X'07'   Reply

X'08'   Request

X'09'   Control

X'0A'   Command

X'0B'   Status

X'0C'   Reserved

Offset to long procedure name. The offset to the field that holds the procedure name.

Procedure name. The name of the procedure that sent or received the message. This field is filled only if the procedure name is less than or equal to 256 bytes in length.

Reserved. An ignored field.

Time stamp. The time when the event occurred.

Trace record type. The message handler program that traced the data:

X'03'   SNDMSG

X'04'   Default exception handler

X'05'   Send status message

X'06'   Reply to SNGMSG

Format of Command Analyzer Data Trace Record

Offset			
Dec	Hex	Type	Field
0	0	CHAR(1)	Data source
1	1	CHAR(8)	Time stamp
9	9	CHAR(10)	Command name
19	13	CHAR(10)	Command library name
29	1D	CHAR(10)	Command processing program name
39	27	CHAR(10)	Command processing program library name

Field Descriptions

Command name. The name of the command that was called.

Command library name. The name of the library containing the command that was called.

Command processing program library name. The name of the library the command processing program is in.

Command processing program name. The name of the command processing program.

Data source. The function the data is from:

X'07'   Command processor

X'08'   Call program

X'0A'   Control language compiler

Time stamp. The time when the event occurred.

Format of Other Data Trace Record

Not all data trace records are formatted for a specific function.

Offset			
Dec	Hex	Type	Field
0	0	CHAR(1)	Data source
1	1	CHAR(8)	Time stamp
9	9	CHAR(300)	Data

Field Descriptions

Data. The data that is being traced.

Data source. The function the data is from.

Time stamp. The time when the event occurred.

Format of Suspend Trace Record

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Data source
1	1	CHAR(8)	Time stamp

Field Descriptions

Data source. The function that was suspended.

Time stamp. The time when the event occurred.

Format of Resume Trace Record

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(1)	Data source
1	1	CHAR(8)	Time stamp

Field Descriptions

Data source. The function was that resumed.

Time stamp. The time when the event occurred.

Exit program introduced: V3R1