System i

Programming
Journal and Commit APIs

Version 6 Release 1

System i

Programming
Journal and Commit APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in "Notices," on page 183.

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

Contents

Journal and Commit APIs

Journaling allows you to specify objects that you want to protect for recovery purposes. It also provides an audit trail for object changes. Journaling provides an audit or activity trail for other objects either through system operations or user actions. The journal APIs allow you to do the following tasks:

- Obtain information about some of the journal's attributes or the journal receiver's attributes.
- Obtain journal information based on the journal identifier.
- Send an entry to specified journal.
- Add, remove, activate, and inactivate remote journals.
- Start and stop journaling.

Commitment control allows you to define and process changes to resources, such as database files or tables, as a single logical unit of work. Commitment control uses the journaling facility to provide for logical units of work. The commitment control APIs allow you to do the following tasks:

- Add and remove your own resources to be used during system commit or rollback processing.
- Retrieve information about the commitment control environment.
- Change commitment control options.
- Put a commitment definition into rollback-required state.

For additional information, see these topics:

- "Journaling for Journal and Commit APIs" on page 176
- "Commitment Control for Journal and Commit APIs" on page 177

If you plan to use the APIs described in this topic, you must understand the Commitment control and Journal management topic collections. These topic collections include information about remote journaling, remote journaling support, and a complete description of the information contained in journal entries that the system sends and all the possible journal codes and entry types.

The journal and commit APIs are:

- "Add Commitment Resource (QTNADDCR) API" on page 3 (QTNADDCR) adds an API commitment resource to the current commitment definition.
- "Add Remote Journal (QjoAddRemoteJournal) API" on page 12 (QjoAddRemoteJournal) associates a remote journal on the target system, as identified by the relational database directory entry, with the specified journal on the source system.
- "Change Commitment Options (QTNCHGCO) API" on page 17 (QTNCHGCO) changes the commitment control options for the current commitment definition.
- "Change Journal Recovery Count (QJOCHRVC) API" on page 22 (QJOCHRVC) how often changes to journaled objects are forced to auxiliary storage.
- "Change Journal State (QjoChangeJournalState) API" on page 23 (QjoChangeJournalState) changes the journal state of local and remote journals.
- "Clear LU6.2 Partners (QTNCLRLU) API" on page 32 (QTNCLRLU) clears LU6.2 syncpoint LOG partners known to the system.
- "Delete Pointer Handle (QjoDeletePointerHandle) API" on page 34 (QjoDeletePointerHandle) deletes the specified pointer handle.
- "End Journal (QjoEndJournal) API" on page 35 (QjoEndJournal) ends journaling for the specified object.
- "Materialize Journal Port Attributes (QusMaterializeJournalPortAttr) API" on page 42 (QusMaterializeJournalPortAttr) retrieves some of the current attributes of a journal.

- "Materialize Journal Space Attributes (QusMaterializeJournalSpaceAttr) API" on page 43 (QusMaterializeJournalSpaceAttr) retrieves some of the current attributes of a journal receiver.
- "Remove Commitment Resource (QTNRMVCR) API" on page 43 (QTNRMVCR) removes an API commitment resource from the current commitment definition.
- "Remove Remote Journal (QjoRemoveRemoteJournal) API" on page 44 (QjoRemoveRemoteJournal) disassociates a remote journal on the target system, as identified by the relational database directory entry, from the specified journal on the source system.
- "Replay Database Operation (QDBRPLAY) API" on page 47 (QDBRPLAY) replays a database operation from a single journal entry.
- "Replay Journal Entry (QjoReplayJournalEntry) API" on page 53 (QjoReplayJournalEntry) replays a single journal entry.
- "Retrieve Commitment Information (QTNRCMTI) API" on page 58 (QTNRCMTI) gets information about the current commitment definition.
- "Retrieve Journal Entries (QjoRetrieveJournalEntries) API" on page 63 (QjoRetrieveJournalEntries) provides access to journal entries.
- "Retrieve Journal Identifier Information (QJORJIDI) API" on page 101 (QJORJIDI) gets information about a specific journal identifier (JID) for a specified journal.
- "Retrieve Journal Information (QjoRetrieveJournalInformation) API" on page 105 (QjoRetrieveJournalInformation) provides access to journal-related information to help manage a journal environment, including a remote journal environment.
- "Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API" on page 131 (QjoRtvJrnReceiverInformation) provides access to journal-receiver-related information to help manage a journal environment, including a remote journal environment.
- "Return LU6.2 Partners (QTNRTNLU) API" on page 142 (QTNRTNLU) returnss LU6.2 syncpoint LOG partners known to the system.
- "Rollback Required (QTNRBRQD) API" on page 143 (QTNRBRQD) puts the current commitment definition into a rollback-required state.
- "Send Journal Entry (QJOSJRNE) API" on page 144 (QJOSJRNE) writes a single journal entry to a specific journal.
- "Start Journal (QjoStartJournal) API" on page 151 (QjoStartJournal) starts journaling for the specified object.

The journal and commit exit programs are:

- "Change Journal Receiver Exit Program" on page 160 (QIBM_QJO_CHG_JRNRCV) is called when a journal receiver has been detached from a journal.
- "Commitment Control Exit Program" on page 164 is called during commitment control operations after an API commitment resource is added to a commitment definition. The commitment control operations pass specific information to the exit program.
- "Delete Journal Receiver Exit Program" on page 171 (QIBM_QJO_DLT_JRNRCV) is called when a journal receiver is to be deleted by any method. For example, the exit program will be called when the user runs the Delete Journal Receiver (DLTJRNRCV) command, or when the system attempts to delete a journal receiver because the journal has the DLTRCV(*YES) attribute specified.

Top | APIs by category

APIs

These are the APIs for this category.

Add Commitment Resource (QTNADDCR) API

Required Parameter Group:

1	Resource handle	Output	Binary(4)
2	Resource name	Input	Char(10)
3	Qualified commitment control exit program name	Input	Char(20)
4	Commitment control exit program information	Input	Char(80)
5	IPL and ASP device vary on processing option	Input	Char(1)
6	Error code	I/O	Char(*)

Optional Parameter Group 1:

7	Add resource options	Input	Char(*)

Optional Parameter Group 2:

8	Current savepoint number	Output	Binary(4)

Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes

The Add Commitment Resource (QTNADDCR) API adds an API commitment resource to a commitment definition. When the resource has been added, the specified exit program is called during commitment control operations performed for the commitment definition until the resource is removed. Once an API commitment resource is added, it must be removed with the Remove Commitment Resource (QTNRMVCR) API before commitment control can be ended for the commitment definition, unless activation-group-level commitment definitions are used. Activation-group-level commitment definitions for nondefault activation groups are automatically ended by the system and any API commitment resources are implicitly removed when the activation group is ended. See "Remove Commitment Resource (QTNRMVCR) API" on page 43 for more information about this API.

To have several API commitment resources at once, you must use this API to add each resource, one at a time. This API does not check for duplicate resource names or duplicate commitment control exit programs.

API commitment resources are considered either one-phase or two-phase. One-phase API commitment resources cannot be registered with any remote resource. One-phase resources are called once during both commit and rollback processing. Two-phase resources are optionally called three times for commit processing and twice for rollback processing. Optionally, two-phase resources may also be called to reacquire their locks during IPL and ASP device vary on. IPL and ASP device vary on recovery may need to take place after the IPL or vary on finishes and resources that are not locked may not be able to be recovered. For more information about one-phase and two-phase API commitment resources, see the Journal management topic collection.

For each API commitment resource that is added, and specified not to be called during both the classify and prepare phases, a single call is made to the associated exit program by commit or rollback processing. For each two-phase resource added and specified to be called during both the classify and prepare phases, the associated exit program is called three times for commit processing and twice for rollback processing. During the first call (or classify call), the exit program should place its conversations in protected states and force any buffered data. During the second call (or prepare call), the exit program must place its resources in a state where they can be committed, rolled back, or recovered from a system failure. The prepare call is made only for commit processing, not rollback processing. During the third call, the exit program is told to commit or roll back its resources.

A journal name can be specified when the API commitment resource is added to associate a journal with the resource. If specified, the journal must not be a remote journal. The resource can use this journal to

permanently store information that may be needed to commit, rollback, or reacquire locks on the resource. This journal can be used in a manner similar to the way the database uses journals to keep track of record-level I/O. When the commitment control exit program is called to commit or roll back the resource or to reacquire locks during IPL, the commit cycle identifier of the current logical unit of work is passed to the program. This commit cycle identifier can be used as a starting or ending point when receiving, retrieving, or displaying entries from the journal.

Exit programs are grouped according to what is specified for the journal name in the add resource options. All exit programs that have been associated with the same journal are grouped together and all exit programs that are not associated with a journal are grouped together. During commit processing the exit programs are called in the order within the group in which they were added to their particular commitment definition. During rollback processing the exit programs are called in the reverse order. All calls to API commitment resources are made before record-level I/O operations are processed.

For more information about the exit program and information that is passed to it, see "Commitment Control Exit Program" on page 164.

Authorities and Locks

Exit Program Authority
*USE

Exit Program Library Authority
*EXECUTE

Exit Program Lock
*SHRNUP

Journal Authority
*USE

Journal Library Authority
*EXECUTE

Journal Lock
*SHRUPD

Restrictions

You are prevented from adding a commitment resource using this API when:

- Distributed data management (DDM) or distributed relational database is used to update remote resources under commitment control and two-phase commit protocols are not supported at the remote system.

  Note: If remote resources are read-only, the API can be used to register a commitment resource as long as the resource is compatible with remote resources. See the Add resource options (page 6) parameter for more details.

  Note: You can use the "Retrieve Commitment Information (QTNRCMTI) API" on page 58 to retrieve information about what type of commitment control resources are currently associated with the currently active commitment definition for the program making the retrieve request.
- Commitment control is not active for the program when making the request to add a commitment resource.
- Commitment control cannot get a shared-no-update (*SHRNUP) lock on the commitment control exit program.
- Commitment control cannot get a shared-for-update (*SHRUPD) lock on the journal associated with this resource. This is a restriction only if a journal is specified to be associated with the resource.
- A commitment control operation is currently in progress for the commitment definition that is to have the commitment resource added.

- The checkpoint processing for a save-while-active function is in progress in another job, when you specify the option to allow normal save processing or specify the default (N).
- » The checkpoint processing for an independent ASP quiesce is in progress on the system, when you specify the option to allow independent ASP quiesce or specify the default (N). «

In addition to the preceding restrictions, you are prevented from adding a one-phase API commitment resource when any remote resources exist for the commitment definition. Adding a resource is also disallowed when incompatible option values are specified.

In all other instances, the API commitment resource is added to the commitment definition.

Once a resource has been added to a commitment definition, the process must not change the authorities to the commitment control exit program or delete the exit program.

Required Parameter Group

Resource handle
OUTPUT; BINARY(4)

An identifier made up of an arbitrary number returned by the API and used to identify the commitment resource for subsequent operations, such as the Remove Commitment Resource (QTNRMVCR) API.

Resource name
INPUT; CHAR(10)

The name that identifies this commitment resource. It is used, for example, in some error messages associated with the commitment control exit programs.

Qualified commitment control exit program name
INPUT; CHAR(20)

The name of the commitment control exit program to be called from the commitment control operations and the library in which it is located. The exit program must exist when this API is called.

The exit program must reside on the same ASP as the commitment definition to which the API commitment resource is added. If the exit program can be called during ASP device vary on processing, it may also reside on the system ASP.

The first 10 characters of this name contain the program name, and the second 10 characters contain the library name. The special values supported for the library name are *LIBL and *CURLIB.

Note: The special values *LIBL and *CURLIB apply only to the time the resource is added. For example:

1. The API user specifies PROGRAMA in *CURLIB when a commitment resource is added. LIBRARYA is the *CURLIB when the resource is added.
2. After the resource addition, *CURLIB is changed to LIBRARYB, which also happens to contain a PROGRAMA.
3. The commit operation occurs and PROGRAMA in LIBRARYA is called, not PROGRAMA in LIBRARYB.

The user of this API must supply this exit program. The considerations for coding this exit program, as well as the information that the commitment control operations pass to this exit program, are described in the “Commitment Control Exit Program” on page 164.

Commitment control exit program information
INPUT; CHAR(80)

Data to be passed directly to the commitment control exit program. This may be any data that is needed by the exit program, such as a reference to an object or area to be used by the exit program. This may be any type of data, including pointers. However, if pointers are used, this field must be on a 16-byte boundary.

Pointers provide better performance than if this parameter were an object name. Resolving to an object on every commit or rollback operation degrades performance. However, pointers to data residing on an ASP may become not usable if the ASP is no longer available.

If the exit program is to be called during IPL or ASP device vary on processing, the information passed-in or pointed-to by this parameter must not be temporary. That is, the information referred to and used by the exit program must persist across an IPL and ASP device vary on.

IPL and ASP device vary on processing option
INPUT; CHAR(1)

Whether the commitment control exit program will be called during any commitment control processing that occurs during IPL and/or ASP device vary on recovery processing for the commitment definition.

N If the API resource is in the commitment definition when the system ends abnormally, the commitment control exit program is not called during the IPL recovery processing for the commitment definition.

Y If the API resource is in the commitment definition when the system ends abnormally, the commitment control exit program is called during the IPL recovery processing for the commitment definition.

V If the API resource is in the commitment definition when the ASP device ends abnormally, the commitment control exit program is called during the ASP device vary on recovery processing for the commitment definition.

B If the API resource is in the commitment definition when the system or ASP device ends abnormally, the commitment control exit program is called during the IPL and ASP device vary on recovery processing for the commitment definition.

Note: When called during IPL or ASP device vary on, the exit program runs under the same user profile that originally added the commitment resource.

The order in which commitment definitions are processed during IPL or ASP device vary on recovery processing is not predictable. However, for each particular commitment definition, the commitment control exit programs are grouped according to what was specified for the associated journal name when they were added with the QTNADDCR API. All exit programs that were associated with the same journal are grouped together, and all exit programs that were not associated with a journal are grouped together. If a commit operation is being finished during IPL or ASP device vary on recovery, the programs within each group are called in the order they were added. If a rollback is being performed, the programs are called in reverse order.

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

Add resource options
INPUT; CHAR(*)

A structure of input options. See “Input Options Structure” on page 7 for the format of the options and a description of the individual options.

When this parameter is specified, the optional parameters will be passed to the commitment control exit program when it is called. See the “Commitment Control Exit Program” on page 164 for more information.

If the add resource options parameter is left out, the API commitment resource is assumed to be a one-phase API commitment resource. The other options are ignored and the options are not passed to the exit program.

Optional Parameter Group 2

Current savepoint number
OUTPUT; BINARY(4)

An identifier of the savepoint assigned to the savepoint name. This identifier may not increment by 1 because of internally created savepoints.

Input Options Structure

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Structure length
4	4	CHAR(20)	Qualified journal name
24	18	CHAR(1)	Resource protocol
25	19	CHAR(1)	Call for classify
26	1A	CHAR(1)	Call for prepare
27	1B	CHAR(1)	Call for rollback required
28	1C	CHAR(1)	Call for reacquiring locks during IPL or ASP device vary on
29	1D	CHAR(1)	Last agent
30	1E	CHAR(1)	Allow normal save processing
31	1F	CHAR(1)	Savepoint compatible
32	20	CHAR(1)	Call for setting a savepoint
33	21	CHAR(1)	Call for rollback to a savepoint
34	22	CHAR(1)	Call for release a savepoint
35	23	CHAR(1)	Allow independent ASP quiesce

Field Descriptions

Allow normal save processing. Whether the registration of this API commitment resource allows save processing to perform normally.

If multiple API commitment resources are to be registered, they all must specify Y in order to prevent poor performance of save-while-active processing.

Valid values for this option are:

N This resource does not allow all save requests to perform normally.
- All save operations that are attempted from the job in which the resource is registered are rejected. The resource must be removed before a save can be performed in the job.
- Save operations that are attempted from other jobs, and that specify save-while-active, wait for this resource to be at a commitment boundary. A commit or rollback must be performed for the job in which the resource is registered before the save-while-active will be allowed in the other job.
- Save operations that are attempted from other jobs, and that do not specify save-while-active, perform normally. They do not wait for this resource to be at a commit boundary.

Y This resource will allow all save requests to perform normally.

Note: If the optional parameter group is not specified, this resource does not allow all save requests to perform normally.

» **Allow independent ASP quiesce.** Whether the registration of this API commitment resource allows independent ASP to be quiesced.

Valid values for this option are:

N This resource does not allow independent ASP to be quiesced.

Y This resource will allow independent ASP to be quiesced.

Notes:

1. If the optional parameter group is not specified, this resource does not allow ASP to be quiesced.
2. Allow independent ASP quiesce options only apply to the registration of the resource. If a journal entry is being sent on behalf of the API resource, it will be held at that point if a quiesce is active, regardless of the specified quiesce option values.

«

Call for classify. Whether the commitment control exit program should be called during the classify phase of a commit or rollback. The commitment control exit program is called during the classify phase to use protected conversations and force any buffered data.

Valid values for this option are:

N Do not call the commitment control exit program during the classify phase of commit or rollback processing.

Y Call the commitment control exit program during the classify phase of commit or rollback processing.

Note: One-phase API commitment resources cannot be called for classify.

Call for prepare. Whether the commitment control exit program should be called during the prepare phase of a commit. The commitment control exit program is called during the prepare phase of the commit to put its resources in a position to either commit, rollback, or recover from a system failure. The commitment control exit program is also given a chance to vote whether this logical unit of work should commit, rollback, or that the resources associated with this commitment control exit program have not been changed. If the resources have not been changed then the exit program can choose not to be called during the second phase of the commit. This is commonly referred to as voting read-only.

Voting is done by setting flags in the parameter structure which is passed to the commitment control exit program when it is called.

Valid values for this option are:

N Do not call the commitment control exit program during the prepare phase of commit processing. Commit processing assumes the vote is to commit the logical unit of work.

Y Call the commitment control exit program during the prepare phase of commit processing.

Notes:

1. One-phase API commitment resources cannot be called for prepare.
2. Two-phase API commitment resources with a Last agent option value of Y cannot be called for prepare.

Call for reacquiring locks during IPL or ASP device vary on. Whether the commitment control exit program should be called during IPL or ASP device vary on if locks need to be reacquired. Under some circumstances, IPL or ASP device vary on recovery cannot be completed for this resource until after the

IPL or ASP device vary on is complete. A call can be made to the commitment control exit program so that any locks which were protecting this resource can be reacquired before the IPL or ASP device vary onis complete.

It is the responsibility of the application that added the resource to keep track of which locks need to be reacquired during IPL or ASP device vary on.

Valid values for this option are:

N Do not call the commitment control exit program during IPL to reacquire locks.
Y Call the commitment control exit program during IPL to reacquire locks.
V Call the commitment control exit program during ASP device vary on to reacquire locks.
B Call the commitment control exit program during both IPL and ASP device vary on to reacquire locks.

Notes:

1. One-phase API commitment resources cannot be called to reacquire locks during IPL.
2. Two-phase API commitment resources with an IPL processing option value of N cannot be called to reacquire locks during IPL. If the optional parameter group is not specified, the commitment control exit program is not called during IPL to reacquire locks.

Call for rollback required. Whether the commitment control exit program should be called if the commitment definition to which this resource was added is put in a rollback-required state. When a commitment definition is placed in a rollback-required state, the use of protected resources is not allowed until the commitment definition is rolled back. The commitment control exit program should take the necessary action so that the API resources registered cannot be used until a rollback is done.

Valid values for this option are:

N Do not call the commitment control exit program when the commitment definition is put into a rollback-required state.
Y Call the commitment control exit program when the commitment definition is put into a rollback-required state.
Note: One-phase API commitment resources cannot be called for rollback-required state.

Call for rollback to a savepoint. Whether the commitment control exit program should be called when rollback to savepoint is requested for the commitment definition to which this resource was added.

Valid values for this option are:

N Do not call the commitment control exit program when rollback to savepoint is requested for the commitment definition.
Y Call the commitment control exit program when rollback to savepoint is requested for the commitment definition.

Call for release a savepoint. Whether the commitment control exit program should be called when release savepoint is requested for the commitment definition to which this resource was added.

Valid values for this option are:

N Do not call the commitment control exit program when release savepoint is requested for the commitment definition.
Y Call the commitment control exit program when release savepoint is requested for the commitment definition.

Call for setting a savepoint. Whether the commitment control exit program should be called when a savepoint is established for the commitment definition to which this resource was added.

Valid values for this option are:

N Do not call the commitment control exit program when a savepoint is established for the commitment definition.
Y Call the commitment control exit program when a savepoint is established for the commitment definition.

Last agent. Whether this commitment resource should be called as the last agent. The last agent is called after all resources have been prepared and before any resources have been committed. This resource will make the decision about whether this logical unit of work commits or rolls back.

Specifying an API commitment resource to be called as the last agent could cause incompatibilities between applications. It will also cause the logical unit of work to be rolled back if a last agent cannot be selected.

A single call will be made to the commitment control exit program if it is the last agent. This exit program must commit or roll back its resources and then inform commitment control of what it did through the Commit Vote return field.

If the call to the exit program fails (an exception is returned) or if the system fails during the call to the exit program, the logical unit of work will be committed or rolled back according to the Action if problems commitment option. The Action if problems commitment option can be changed with the Change Commitment Options (QTNCHGCO) API.

There can be only one last agent per commitment definition. Escape message CPF8369 is issued with reason code 13 if an attempt is made to add a last agent commitment resource when one is already registered.

Escape message CPF8369 is issued with reason code 7 if an attempt is made to add a last agent commitment resource when the Last agent permitted commitment option is set to N. The Last agent permitted commitment option can be changed with the Change Commitment Options (QTNCHGCO) API.

Valid values for this option are:

N This resource should not be called as the last agent.
Y This resource should be called as the last agent.
Notes:

1. One-phase API commitment resources cannot be called as the last agent.
2. Two-phase API commitment resources with a Call for prepare option value of Y cannot be called as the last agent.

Qualified journal name. The name of the journal that will be associated with this resource. The first 10 characters of this name contain the journal name, and the second 10 characters contain the library name. The special value *NONE is supported for the journal name if no journal is to be associated with this API resource. The special value *DFTJRN specifies that the default journal specified when the commitment definition was started should be associated with this commitment resource. The special value of *DFTJRN will be substituted by the journal name specified when the commitment definition was started. If either of these special values are specified, the library name is ignored. The special values supported for the library name are *LIBL and *CURLIB.

Note: The special values *LIBL and *CURLIB apply only to the time the resource is added. For example:

1. The API user specifies JOURNALA in *CURLIB when a commitment resource is added. LIBRARYA is the *CURLIB at the time the resource is added.
2. After the resource addition, *CURLIB is changed to LIBRARYB, which also happens to contain a JOURNALA.
3. The commit operation occurs using JOURNALA in LIBRARYA, not JOURNALA in LIBRARYB.

Entries can be placed in the specified journal which could be used later by the commitment control exit program to recover resources or reacquire locks. See "Send Journal Entry (QJOSJRNE) API" on page 144 (QJOSJRNE) API for information about sending journal entries. If a commit cycle has not been started for the journal during the current logical unit of work, one is started when the user requests to include the commit cycle identifier when sending a journal entry using the QJOSJRNE API. The commit cycle identifier will be passed to the commitment control exit program and this commit cycle identifier can be used as a starting or ending point when receiving, retrieving, or displaying entries from the journal. The CL commands RCVJRNE and RTVJRNE can be used to receive and retrieve journal entries. The DSPJRN command can display, print to a spool file, or put to an output file the journal entries.

If the optional parameter group is not specified, no journal will be associated with the API resource.

Resource protocol. Whether this API commitment resource is a one-phase or a two-phase commitment resource. One-phase commitment resources are not compatible with any remote commitment resources and cannot be called to classify, prepare, reacquire locks during IPL, or as the last agent. Two-phase commitment resources can optionally be called to classify, prepare, reacquire locks during IPL, or as the last agent.

Valid values for this option are:

1 This is a one-phase API commitment resource.
2 This is a two-phase API commitment resource.

One-phase API commitment resources do not have the ability to fully protect themselves against a remote resource failure.

Savepoint compatible. Whether the commitment control resource is compatible with savepoints.

Valid values for this option are:

N The commitment control resource cannot be registered while a savepoint is in effect.
Y The commitment control resource can be registered while a savepoint is in effect. Also, the exit program may be optionally called when a savepoint is set, released or rolled back.

Structure length. The length of the input structure provided. To provide a journal without the remaining options, specify 24 for the structure length. If a structure length greater than 24 is specified, and no journal is desired, specify special value *NONE for the journal.

Valid values for this option are:

24 Only the journal is provided as an option.
31 Only the journal and two-phase commit related options are provided.
35 All options except allow independent ASP quiesce are provided.
36 All options are provided.

Usage Notes

This API is always threadsafe. However, since the specified commitment control exit program is called during commitment control operations performed in the same job, the exit program must also be threadsafe if the API is used in a multithreaded job.

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.
CPF705A E	Operation failed due to remote journal.
CPF836A E	Value &1 not valid for option &2.
CPF836D E	Resource name &1 not valid.
CPF8367 E	Cannot perform commitment control operation.
CPF8369 E	Cannot place API commitment resource under commitment control; reason code &1.
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.
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 &1in library &2 ended. Reason code &3.

API introduced: V2R2

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Add Remote Journal (QjoAddRemoteJournal) API

Required Parameter Group:

1	Qualified journal name	Input	Char(20)
2	Relational database directory entry	Input	Char(18)

Omissible Parameter Group:

3	Request variable	Input	Char(*)
4	Length of request variable	Input	Binary(4)
5	Format name of request variable	Input	Char(8)
6	Error code	I/O	Char(*)

Default Public Authority: *USE
Service Program: QJOURNAL
Header File: QSYSINC/H.QJOURNAL
Threadsafe: No

The Add Remote Journal (QjoAddRemoteJournal) API associates a remote journal on the target system, as identified by the relational database directory entry, with the specified journal on the source system. The journal on the source system may be either a local journal or another remote journal. A maximum of 255 remote journals may be associated with a single journal on a source system.

When adding a remote journal to a source journal, the remote journal is created on the target system using a combination of the attributes from the source journal and the input parameters provided on this API. The library that the remote journal will be created in must already exist on the target system prior to

this API being called on the source system. When created by this API, the remote journal will be created with a journal type of *REMOTE and the remote journal will not have an attached journal receiver.

When adding the remote journal, the remote journal can either be created into the same named library as that of the source journal or into a redirected library on the target system. A redirected library provides a means for remote journals and any of their associated journal receivers to reside in different named libraries on the target system from the corresponding local journal and journal receivers on the local system. When specified, all validation for the journal library on the target system will be performed using the redirected library name. Similarly, the journal receivers that will later be created and associated with this remote journal can either reside in the same library as the source journal receivers on the source system, or into a distinct redirected library name on the target system. The journal receiver library redirection, if desired, must be specified when the remote journal is added using this API.

When adding a remote journal on a target system, two remote journal types can be specified, *TYPE1 and *TYPE2. The remote journal type influences the redirection capabilities, journal receiver restore operations, and remote journal association characteristics. See the Journal management topic collection for detailed descriptions of the differences.

If the specified journal already exists on the target system, the journal can be associated with the source journal, but only if the journal is of type *REMOTE, the remote journal type matches the specified journal type, and the journal was previously associated with this same source journal. Also, the journal may or may not have an attached journal receiver.

After the remote journal has been successfully added on the target system, the remote journal will have a journal state of *INACTIVE. A journal state of *INACTIVE for a remote journal means that the remote journal is currently not receiving journal entries from its source journal on the source system. The Change Remote Journal (CHGRMTJRN) command or the Change Journal State (QjoChangeJournalState) API is used to activate a remote journal and start the replication of journal entries from the source journal to the remote journal.

Once a remote journal has been added to a journal, the journal receiver which was attached at that time on the source system, and any journal receivers attached after that time on the source system, will be protected from deletion if all journal entries for a given journal receiver have not yet been replicated to the remote journal. This protection ends when the remote journal is removed using the Remove Remote Journal (RMVRMTJRN) command or the Remove Remote Journal (QjoRemoveRemoteJournal) API.

Restrictions

The following restrictions apply:

- The Add Remote Journal (QjoAddRemoteJournal) API may only be used from the source system for a local or remote journal.
- A user profile must exist on the target system by the same name as the user profile that is running the Add Remote Journal (QjoAddRemoteJournal) API on the source system.
- When adding a *TYPE1 remote journal to a source journal, the same journal and journal receiver library redirection must be specified that exists for any *TYPE1 remote journals which have already been added to the source journal. A remote journal will always use the redirected library, if any, that is specified for the local journal.

  Note: The only way to change the remote journal library field and the remote journal receiver library field for a *TYPE1 journal is to do all of the following:

  1. Remove all *TYPE1 remote journals.
  2. Change the local journal and attach a new journal receiver.
  3. Delete the remote journal from the target system.
  4. Add the *TYPE1 remote journal, and specify the new library redirection, if any.

- QTEMP cannot be specified for the remote journal library, remote journal receiver library, or remote message queue library.
- A remote journal whose name starts with a Q cannot specify a remote journal library that starts with a Q, unless the remote journal library is QGPL. This is required to prevent collisions between local and remote journals that are used for system functions.
- A *TYPE1 remote journal cannot be added to a *TYPE2 remote journal.
- The remote journal message queue on the remote journal system must be either in the same ASP group as the remote journal, or in the system ASP, or a basic user ASP.
- The remote receiver library and remote journal library on the remote system must both exist in either the system and basic user ASPs or in the same ASP group. They cannot be in two different ASP groups.

Authorities and Locks

Source Journal Authority
*CHANGE, *OBJMGT

Source Journal Library Authority
*EXECUTE

Target Journal Library Authority
*EXECUTE, *ADD

Service Program Authority
*EXECUTE

Source Journal Lock
*EXCLRD

Target Journal Library Lock
*SHRUPD

Target Journal Lock
*SHRUPD

Required Parameter Group

Qualified journal name
INPUT; CHAR(20)

The name of the journal on the source system to which the remote journal is being added, and the library where it resides. The journal on the source system may be either a local journal or another remote journal. The first 10 characters contain the journal name, and the second 10 characters contain the name of the library where the journal is located.

The special values supported for the library name follow:

*LIBL	Library list
*CURLIB	Current library

Relational database directory entry
INPUT; CHAR(18)

The name of the relational database directory entry that contains the remote location name of the target system. This name should match the name of the *LOCAL relational database directory entry on the target system.

Omissible Parameter Group

Request variable
INPUT; CHAR(*)

The request variable structure that describes the input for the Add Remote Journal (QjoAddRemoteJournal) API.

Length of request variable
INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be 102 bytes or greater.

Format name of request variable
INPUT; CHAR(8)

The format ADRJ0100 is the only supported format that is used by this API. See "ADRJ0100 Format" for more information on the ADRJ0100 format.

Error code
I/O; CHAR(*)

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

ADRJ0100 Format

The following table defines the information that may be provided for format ADRJ0100 when you add a remote journal.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(20)	Qualified remote journal name
20	14	CHAR(10)	Remote journal receiver library
30	1E	CHAR(1)	Remote journal type
31	1F	CHAR(20)	Qualified journal message queue
51	33	CHAR(1)	Delete receivers
52	34	CHAR(50)	Text
102	66	CHAR(2)	Reserved
104	68	BINARY(4)	Delete receivers delay

Field Descriptions

Delete receivers. Whether the system deletes the target journal receivers when they are no longer needed or keeps them on the target system for the user to delete after they have been detached by the target system. If this field is not provided or is blank, a value of 0 is assumed. A value is only set for a journal that is created on the target system.

The possible values are:

0 The target journal receivers are not deleted by the target system.
1 The target journal receivers are deleted by the target system.

Delete receivers delay. The number of minutes (from 1 to 1440) the target system waits to retry deleting a target journal receiver if it cannot be allocated on the target system. If this field is not provided, a value of 10 minutes is assumed. A value is set for a journal that is created on the target system only.

The possible values are:

Number The number of minutes to delay, from 1 through 1440.

Qualified journal message queue. The qualified name of the message queue that is associated with this remote journal. If this field is not provided or is blank, a value of QSYSOPR QSYS is assumed. A value is only set for a journal that is created on the target system.

The possible values are:

QSYSOPR QSYS The message is sent to the QSYSOPR message queue in library QSYS.

Journal message queue The name of the message queue to which the remote journal messages are sent on the target system. If this message queue is not available when a message is to be sent, the message is sent to the QSYSOPR message queue. The first 10 characters contain the message queue name, and the second 10 characters contain the name of the library where the message queue is located.

Qualified remote journal name. The qualified name of the remote journal on the target system. The first 10 characters contain the remote journal name; the second 10 characters contain the name of the library where the remote journal is to be located. If this field is not provided or is blank, the resolved qualified journal name is assumed.

If the remote journal type is *TYPE1, then the remote journal name must match the specified journal name. Whether a *TYPE1 or *TYPE2 remote journal is being added, the library name can be any name which will become the redirected journal library name.

Remote journal receiver library. The name of the library for the remote journal receivers on the target system that is associated with this remote journal. If this field is not provided or is blank, the journal receivers are created on the target system in the same library as they exist on the source system.

Remote journal type. The type of remote journal on the target system. The remote journal type influences the redirection capabilities, journal receiver restore operations, and remote journal association characteristics. See the Journal management topic collection for detailed descriptions of the differences. If this field is not provided or is blank, a value of 1 is assumed.

The possible values are:

1 A *TYPE1 remote journal will be added.

2 A *TYPE2 remote journal will be added.

Reserved. The bytes reserved to align binary fields or for future use. This field must be set to hexadecimal zero.

Text. The text that briefly describes the remote journal on the target system. If this field is not provided, a value of all blanks is assumed. A value is only set for a journal that is created on the target system.

The possible values are:

Text No more than 50 characters of text.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C39 E	Value for reserved field not valid.
CPF3C4E E	Value not valid for field &1.
CPF3C90 E	Literal value cannot be changed.
CPF69A4 E	Remote journal &1 in &2 not added.
CPF695A E	Remote journal &1 in &2 not added.
CPF695B E	Remote journal &1 in &2 not added.
CPF695C E	Remote journal &1 in &2 not added.
CPF695D E	Remote journal &1 in &2 not added.
CPF695E E	Remote journal &1 in &2 not added.
CPF695F E	Remote journal &1 in &2 not added.
CPF696A E	Request variable length &1 not valid.
CPF6973 E	Systems not compatible.
CPF6982 E	Relational database directory entry &1 not valid.
CPF6983 E	Remote journal &1 in &2 not added.
CPF6984 E	Remote journal &1 in &2 not added.
CPF6985 E	Remote journal &1 in &2 not added.
CPF6986 E	The request variable parameters are in error.
CPF6987 E	Field value &1 specified incorrectly.
CPF6988 E	Remote journal &1 in &2 not added.
CPF6989 E	Remote journal &1 in &2 not added.
CPF699B E	User profile &8 not found.
CPF6991 E	Remote journal &1 in &2 not added.
CPF70DB E	Remote journal function failed.
CPF70D6 E	Remote journal ended, reason code &6.
CPF701B E	Journal recovery of interrupted operation failed.
CPF7010 E	Object &1 in &2 type *&3 already exists.
CPF7011 E	Not enough storage.
CPF8100 E	All CPF81xx messages could be returned. 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.
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: V4R2

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Change Commitment Options (QTNCHGCO) API

Required Parameter Group:

1	Commitment options	Input	Char(*)
2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Change Commitment Options (QTNCHGCO) API changes the current commitment options. This API will change the commitment options for the commitment definition associated with the activation group for the HLL program that called the API. Commitment options for any other commitment definition will not be affected.

These options affect the operation of the system during a two-phase commit operation. The i5/OS® implementation of two-phase commit is based on the SNA LU 6.2 protocol. Beginning in Version 3 Release 7 Modification 0, the i5/OS (OS/400®) implementation supports the presumed abort architecture and its optimizations that are described in the SNA LU 6.2 protocol. The details of this protocol and the relationships between locations that support the presumed abort architecture and those that support the presumed nothing architecture, should be understood before changing these options. See the Commitment control topic collection, LU 6.2 Reference: Peer Protocols, SC31-6808, and Transaction Programmer's Reference Manual for LU Type 6.2, GC30-3084, for detailed information.

Authorities and Locks

None.

Required Parameter Group

Commitment options
INPUT; CHAR(*)

Error code
I/O; CHAR(*)

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

Commitment Options Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Structure length
4	4	CHAR(1)	Wait for outcome
5	5	CHAR(1)	Action if problems
6	6	CHAR(1)	Vote read-only permitted
7	7	CHAR(1)	Action if ENDJOB
8	8	CHAR(1)	Last agent permitted
9	9	CHAR(1)	OK to leave out
10	A	CHAR(1)	Accept vote reliable

Field Descriptions

Accept vote reliable. Whether this location accepts the vote reliable indicator if it is received from its agents during the prepare wave of a commit operation. The vote reliable indicator indicates that it is unlikely that a heuristic decision will be made at the agent if a failure occurs before the committed wave completes. If an agent sends the vote reliable indicator, and this location accepts it, performance is improved because one communications flow is eliminated and control is returned to the application before the committed wave is completed for that agent. However, if a heuristic decision that causes heuristic damage is made at that agent, the application at this location will not receive an error message if the Accept vote reliable commitment option is set to Y.

The valid options are:

Y The location accepts the vote reliable indicator.
N The location does not accept the vote reliable indicator.
* Do not change the current value.

When the commitment definition was started, the value for this option was set to Y.

Note: If the commitment definition has a Wait for outcome value of Y or wait, or inherits a value of wait, the value of the Accept vote reliable commitment option is ignored, and the system behaves as though the Accept vote reliable commitment option is No.

See the Commitment control topic collection for more information about the vote reliable indicator.

Action if ENDJOB. The action to take for changes associated with this logical unit of work when the job in which the logical unit of work is a part of is ended during a commit operation while the status of resources is in doubt.

The valid options are:

W The system waits to allow the normal processing of the logical unit of work to complete.
R The changes to local resources whose status is in doubt for this logical unit of work will be rolled back.
C The changes to local resources whose status is in doubt for this logical unit of work will be committed.
* Do not change the current value.

When the commitment definition was started, the value for this option was set to W.

Note: Setting this option to R or C may lead to inconsistencies in the database if a job is ended during a commit operation while the status of resources is in doubt.

Action if problems. The action to take if the system receives an unrecognized message or detects damage in the logical unit of work.

The valid options are:

R The changes to local resources associated with this logical unit of work will be rolled back.
C The changes to local resources associated with this logical unit of work will be committed.
* Do not change the current value.

When the commitment definition was started, the value for this option was set to R.

Last agent permitted. Whether a last agent can be selected when one is eligible to be selected during a commit operation. A last agent is eligible to be selected at the location that initiates a commit operation, and at locations that are selected as a last agent by the location that propagates the commit operation to that location.

Performance is usually enhanced when a last agent is selected because fewer interactions between this location and the last agent are required during a commit operation. However, if a communications failure occurs between a location and its last agent during a commit operation, the commit operation will not complete until resynchronization completes, regardless of the value of the Wait for outcome commitment option. Such a failure will be rare, but this option allows the application writer to consider the negative impact of causing the user to wait indefinitely for the resynchronization to complete when such a failure occurs. This should be weighed against the performance enhancement that is provided by last agent optimization during successful commit operations. This consideration would generally be more significant for interactive jobs than for batch jobs.

There is one case where performance is not enhanced when a last agent is selected. If no committable changes have been made at an agent, and the Vote read only permitted commitment option has been set to Y at that agent, then performance would actually be degraded by selecting that agent as a last agent. The decrease in performance occurs because fewer write operations to auxiliary storage are required when the vote read only optimization is used. Therefore, applications written such that no data is changed at all agents during most logical units of work should set the Last agent permitted option to N.

The valid options are:

S The system is allowed to select a last agent at this location.
N The system is not allowed to select a last agent at this location.
* Do not change the current value.

When the commitment definition was started, the value for this option was set to S.

Note: The Last agent permitted commitment option cannot be changed to N if an API commitment resource that is specified to be called as the last agent has already been added to the commitment definition using the Add Commitment Resource (QTNADDCR) API.

OK to leave out. Whether this location indicates that it is OK to leave out during a commit operation initiated at another location. **OK to leave out** means that no communications flows are sent to this location during subsequent commit or rollback operations until a data flow is received from the initiator. Also, control is not returned to the application until the data flow is received. The length of the delay in regaining control depends on the application running at the initiator.

In a client/server environment, setting the OK to leave out to Y at all the servers provides improved performance if data is not sent to all servers during every logical unit of work (LUW). The OK to leave out value is communicated from a server to a client during commit operations. Therefore, changing the OK to leave out value from N to Y does not take effect until after the next commit operation. Likewise, changing the OK to leave out value from Y to N does not take effect until after the commit of the next LUW, during which data has been sent to the server. Note that the OK to leave out value is not communicated during rollback operations.

The valid options are:

Y This location may be left out of subsequent logical units of work.
N This location may not be left out of subsequent logical units of work.
* Do not change the current value.

When the commitment definition was started, the value for this option was set to N.

Structure length. The length of the input structure provided. The minimum valid structure length is 5. If the length indicates that one or more options are not passed, the current value for those options is not changed.

Vote read-only permitted. Whether this location can vote read-only in response to a commit operation initiated at another location. If this location does vote read-only, control is not returned to the application until this location gets a message from the initiating location that indicates a new logical unit of work has started. The indicator flows in the data sent from the initiator of the previous commit operation. The length of the delay in regaining control depends on the application running at the initiator.

See the Commitment control topic collection for more information about when a location will vote read-only.

The valid options are:

N This location is not allowed to vote read-only.
Y This location is allowed to vote read-only.
* Do not change the current value.

When the commitment definition was started, the value for this option was set to N.

Wait for outcome. Whether the system will wait for the outcome of commit or rollback operations.

The valid options are:

Y The system completes resynchronization before allowing the commit or rollback operation to complete.
L Value L has the same effect as value Y when this location is the initiator of the commit or rollback operation. When this location is not the initiator, and the initiator supports the presumed abort protocol, the Wait for outcome value is inherited from the initiator. When this location is not the initiator, and the initiator does not support the presumed abort protocol, value L has the same effect as value Y.
N The system attempts resynchronization once before allowing the commit or rollback operation to complete. If resynchronization fails, it will be completed in a system server job. The application will not be notified of the result of the resynchronization.
U Value U has the same effect as value N when this location is the initiator of the commit or rollback operation. When this location is not the initiator, and the initiator supports the presumed abort protocol, the Wait for outcome value is inherited from the initiator. When this location is not the initiator, and the initiator does not support the presumed abort protocol, value U has the same effect as value N.
* Do not change the current value.

When the commitment definition was started, this option was set to Y.

Notes:

1. The Wait for outcome value has no effect when a failure occurs while a logical unit of work is in doubt if the failure is between this location and the location that owns the commit or rollback decision. This is the case if the LUW state is PREPARED and the failure occurs between this location and the initiator location, or if the LUW state is LAST AGENT PENDING and the failure occurs between this location and the last agent location. In this case, the system always waits for the resynchronization to complete regardless of the Wait for outcome value.
2. If the ENDJOB command is used to end a job while it is waiting for resynchronization to complete, the Wait for outcome value is changed to N so that the job is allowed to end quickly. However, if the logical unit of work is in doubt, the Action if ENDJOB commitment option controls whether the job will be allowed to end before resynchronization completes with the location that owns the commit or rollback decision.
3. Consider the following when setting the Wait for outcome value:
   - When the Wait for outcome value is N, the application will not learn about inconsistencies at other locations in the transaction program network. Inconsistencies can result only if a system operator takes manual intervention due to the failure that caused the resynchronization.
   - When the Wait for outcome value is Y, most of the performance benefits provided by the presumed abort protocol are lost.
   - The Wait for outcome value is most useful when the same value is used at all locations in the transaction program network. If the Wait for outcome value is Y at the initiator but N at an agent, the initiator may not learn the full outcome of the commit or rollback operation because the agent did not wait for it. If the Wait for outcome value is N at the initiator but Y at an agent, the initiator may be forced to wait for the outcome of resynchronization performed by the agent. Therefore, it is recommended that all locations in the transaction program network specify either L or U for the Wait for outcome value. This allows the entire tree to use a consistent value because the initiator's value will be inherited by all locations.

- If the commitment definition has a Wait for outcome value of Y or wait, or inherits a value of Wait, the value of the Accept vote reliable commitment option is ignored, and the system behaves as though the Accept vote reliable commitment option is No.

Restrictions

You are prevented from changing the commitment options using this API when:

- Commitment control is not active for the program when making the request to change the commitment options.
- A commitment control operation is in progress for the current commitment definition whose options are to be changed.

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.
CPF83D5 E	Cannot change commitment options; reason code &1. **Note:** Reason codes for the previous message will include commitment control not being started and value specified for option not valid.
CPF8367 E	Cannot perform commitment control operation.

API introduced: V4R4

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Change Journal Recovery Count (QJOCHRVC) API

Required Parameter Group:

1	Recovery count	Input	Binary(4)
2	Error Code	I/O	Char(*)

Default Public Authority: *EXCLUDE
Threadsafe: Yes

The journal recovery count allows a user to choose between faster abnormal IPL recovery and decreased run time processing. The value specified influences the frequency with which journaled objects are forced to auxiliary storage as those objects are changed. The specified journal recovery count indicates the approximate number of journaled changes that would need to be recovered during journal synchronization for this journal in the event of an abnormal IPL. Specifying a smaller value decreases the number of changes that would need to be recovered from this journal during an abnormal IPL by increasing the frequency with which changed objects are forced. Specifying a larger value increases the number of changes that would need to be recovered for this journal during an abnormal IPL by decreasing the frequency with which changed objects are forced. Changing this value may affect overall system performance as it affects the utilization of auxiliary storage devices.

The operating system is shipped with a system default journal recovery count of 250,000. This API changes the system default journal recovery count for all newly created journals on the system and all existing journals that have the system default (*SYSDFT) specified for their journal recovery count. To set the journal recovery count of a specific journal to a value other than the system default, please refer to the Journal recovery count (JRNRCYCNT) parameter on the Change Journal (CHGJRN) command.

This API changes the recovery count for all journals on the system. The default value for journal recovery count is 250,000.

Restrictions

None

Authorities and Locks

*JOBCTL special authority is required to use this API.

Required Parameter Group

Recovery count

INPUT; BINARY(4)

The new value for the journal recovery count. This must be between 10,000 and 2,000,000,000.

Error code

I/O; CHAR(*)

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

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF69A5 E	Recovery count must be between &2 and &3.
CPF69A6 E	Unable to change journal recovery count.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPF90FF E	*JOBCTL special authority required to do requested operation.

API introduced: V5R3

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Change Journal State (QjoChangeJournalState) API

Required Parameter Group:

1	Qualified journal name	Input	Char(20)
2	Request variable	Input	Char(*)
3	Length of request variable	Input	Binary(4)
4	Format name of request variable	Input	Char(8)

Omissible Parameter Group:

5	Receiver variable	Output	Char(*)
6	Length of receiver variable	Input	Binary(4)
7	Error code	I/O	Char(*)

Default Public Authority: *USE
Service Program: QJOURNAL
Header File: QSYSINC/H.QJOURNAL
Threadsafe: No

The Change Journal State (QjoChangeJournalState) API is used to change the journal state for local and remote journals.

When this API is called from the source system for a local journal, the journal state may be changed from *STANDBY to *ACTIVE. A journal state of *ACTIVE for a local journal indicates that journal entries are allowed to be deposited into the attached journal receiver of the journal. A request to change the journal state of a local journal from *ACTIVE to *INACTIVE is ignored.

When this API is called from the source system for a remote journal that is associated with a source system journal, the remote journal state may be changed from *ACTIVE to *INACTIVE or from *INACTIVE to *ACTIVE. This API also allows additional attributes that are associated with the journal state to be set. For additional details on the other attributes that are associated with the journal state, see "Field Descriptions" on page 28.

When this API is called from the target system for a remote journal that is associated with a source system journal, the journal state may be changed from *ACTIVE to *INACTIVE.

A journal state of *ACTIVE for a remote journal indicates that journal entries can be received from the associated journal on the source system. A journal state of *INACTIVE for a remote journal indicates that journal entries are currently not being received.

Restrictions

The following restrictions apply:

- A user profile must exist on the target system by the same name as the user profile that is running this API on the source system.
- Synchronous delivery mode is not supported when a remote journal is specified for the qualified journal name parameter.
- The journal state of the remote journal to be activated cannot already be *ACTIVE.
- The journal state of the remote journal to be inactivated cannot already be *INACTIVE.
- If the remote journal state is *CTLINACT, then the remote journal cannot be inactivated by specifying a preferred inactivate type of controlled inactivate.
- The remote journal to be activated cannot already be replicating journal entries to other remote journals.
- A journal receiver that was never attached to a journal after Version 4 Release 2 Modification 0 has been installed cannot be replicated.
- Format CJST0100 cannot be used with a journal whose names starts with a Q, in a library that starts with Q, unless that library is QGPL.

Authorities and Locks

Source Journal Authority
*CHANGE, *OBJMGT

Source Journal Library Authority
*EXECUTE

Source Journal Receiver Authority
*USE, *OBJMGT

Source Journal Receiver Library Authority
*EXECUTE

Service Program Authority
*EXECUTE

Source Journal Lock
*SHRUPD

Source Journal Receiver Lock
*SHRRD

Target Journal Lock
*SHRUPD

Target Journal Receiver Lock
*SHRRD

Target Journal Receiver Library Lock
*SHRUPD

Required Parameter Group

Qualified journal name
INPUT; CHAR(20)

For formats CJST0100 and CJST0200 the name of the journal for which the journal state is being changed, and the library in which it resides. For formats CJST0300, CJST0400, and CJST0500 the name of the source journal that is associated with the remote journal for which the journal state is being changed, and the library in which it resides. The first 10 characters contain the journal name, and the second 10 characters contain the name of the library where the journal is located.

The special values supported for the library name follow:

**LIBL*	Library list
**CURLIB*	Current library

Request variable
INPUT; CHAR(*)

The request variable structure that describes the input for the Change Journal State (QjoChangeJournalState) API.

Length of request variable
INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be equal to the length of the input format specified » when using CJST0100 and CJST0300. « Zero must be specified for this parameter when you use format CJST0200. » The length of the request variable must be 58 bytes or greater when using formats CJST0400 and CJST0500. «

Format name of request variable
INPUT; CHAR(8)

The format of the information that is provided as input for the Change Journal State (QjoChangeJournalState) API.

The possible format names follow:

CJST0100	Activate a local journal from the source system. See "CJST0100 Format" on page 26 for more information on the CJST0100 format.
CJST0200	Inactivate a remote journal from the target system. There is no additional information required for format CJST0200.
CJST0300	Inactivate a remote journal from the source system. See "CJST0300 Format" on page 26 for more information on the CJST0300 format.
CJST0400	Activate a synchronously maintained remote journal from the source system. See "CJST0400 Format" on page 27 for more information on the CJST0400 format.

CJST0500 Activate an asynchronously maintained remote journal from the source system. See "CJST0500 Format" on page 27 for more information on the CJST0500 format.

Omissible Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The receiver variable that is to receive output from the API. The size of the area to receive the output can be smaller than the output returned for the format requested as long as the length of receiver variable parameter is specified correctly. Only format CJST0300 returns output. If this parameter is omitted, the length of receiver variable parameter must also be omitted. If this parameter is specified, the length of receiver variable parameter must also be specified.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable, in bytes. 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 parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. For formats other than CJST0300 this value must be 0 or the parameter must be omitted. For format CJST0300 this value must be greater than or equal to 8 or the parameter must be omitted. If this parameter is omitted, the receiver variable parameter must also be omitted.

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.

CJST0100 Format

The following table defines the information required for format CJST0100 to activate a local journal from the source system.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	New journal state

CJST0300 Format

The following table defines the information required for format CJST0300 to inactivate a remote journal from the source system.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(18)	Relational database directory entry
18	12	CHAR(20)	Remote journal
38	26	CHAR(1)	Preferred inactivate type

The following table defines the information returned in the receiver variable for format CJST0300 after a remote journal has been inactivated:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(18)	Relational database directory entry
26	1A	CHAR(20)	Remote journal
46	2E	CHAR(1)	Preferred inactivate type
47	2F	CHAR(1)	Inactivate type
48	30	CHAR(10)	Inactivate journal receiver name
58	3A	CHAR(10)	Inactivate journal receiver library
68	44	BINARY(4)	Inactivate sequence number
72	48	CHAR(20)	Inactivate sequence number - long

CJST0400 Format

The following table defines the information required for format CJST0400 to activate a synchronously maintained remote journal from the source system.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(18)	Relational database directory entry
18	12	CHAR(20)	Remote journal
38	26	CHAR(20)	Starting journal receiver
» 58	3A	CHAR(1)	Validity checking
59	3B	CHAR(1)	Reserved
60	3C	BINARY(4)	Synchronous sending time-out
64	40	CHAR(8)	Node identifier
72	48	BINARY(4)	Offset to internet address array
76	4C	BINARY(4)	Number of internet addresses
80	50	CHAR(176)	Reserved
*	*	Array(*) of CHAR(45)	Internet address «

CJST0500 Format

The following table defines the information required for format CJST0500 to activate an asynchronously maintained remote journal from the source system.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(18)	Relational database directory entry
18	12	CHAR(20)	Remote journal
38	26	CHAR(20)	Starting journal receiver
» 58	3A	CHAR(1)	Validity checking
59	3B	CHAR(1)	Reserved
60	3C	BINARY(4)	Sending task priority

Offset		Type	Field
Dec	Hex		
64	40	CHAR(8)	Node identifier
72	48	BINARY(4)	Offset to internet address array
76	4C	BINARY(4)	Number of internet addresses
80	50	CHAR(176)	Reserved
*	*	Array(*) of CHAR(45)	Internet address «

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.

Inactivate journal receiver library. The library of the journal receiver that contains the inactivate sequence number. This field will be blank if the system was unable to determine the library name.

Inactivate journal receiver name. The name of the journal receiver that contains the inactivate sequence number. This field will be blank if the system was unable to determine the journal receiver name.

Inactivate sequence number. If a controlled inactivate was initiated, this is the sequence number of the last journal entry that was queued for replication before the API was called. If an immediate inactivate was performed, this is the sequence number of the last journal entry that was replicated to the remote journal. This field will be 0 if the system was unable to determine the last entry that would have been, or was, replicated.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will be in the Inactivate sequence number - long field.

Inactivate sequence number - long. The same field as Inactivate sequence number except the information is in a Char(20) field that is treated as Zoned(20,0).

Inactivate type. How the replication of journal entries was actually ended.

The possible values follow:

0 A controlled inactivate of journal entry replication is being performed. All journal entries already queued to be sent from the source system to the target system will be replicated before the inactivate operation completes.

1 An immediate inactivate of journal entry replication was performed. The system did not continue to replicate queued journal entries before inactivating the remote journal.

» **Internet address.** Up to four internet addresses for data port services to use when communicating to the target system. If you desire to use data port services as an alternate communication method to the target system, you must specify the name of the cluster node identifier and up to four internet addresses to the target system. Configuring clustering and data port services with multiple communication lines can provide better communications resiliency and higher data throughput. The user is responsible for configuring the TCP/IP internet addresses. The internet addresses must be configured between the source system and target system prior to activating the remote journal. See the Configuring TCP/IP topic to configure up to four distinct TCP/IP routes between the source system and the target system. Any

duplicate addresses will be ignored. If this field is not provided, the internet addresses will be set to blanks and the relational database directory entry will be used for all communications to the target system. «

New journal state. Whether the depositing of journal entries into the local journal should be activated.

The possible values follow:

1 *ACTIVE

» **Node identifier.** The node identifier for data port services to use when identifying the target system in a cluster environment. If you desire to use data port services as an alternate communication method to the target system, you must specify the name of the cluster node identifier and up to four internet addresses to the target system. Configuring clustering and data port services with multiple communication lines can provide better communications resiliency and higher data throughput. The user is responsible for configuring a cluster environment with the target system as one of its nodes. The cluster node for the target system must be active prior to activating the remote journal. For more information about setting up a cluster, see the Configuring clusters topic. If this field is not provided, a value of *NONE will be assumed and the relational database will be used for all communications to the target system.

The possible values follow:

**NONE* The remote journal environment will not use data port services. The relational database will be used for all communications to the target system.

Node-Identifier The cluster node identifier for the target system.

Number of internet addresses. The number of addresses specified in the internet address array. If this field is not provided, a value of 0 will be assumed.

Offset to internet address array. The byte offset from the beginning of the request variable to the first internet address. If this field is not provided, a value of 0 will be assumed. «

Preferred inactivate type. How the replication of journal entries should be ended.

The possible values follow:

0 A controlled inactivate of journal entry replication should be performed. A **controlled inactivate** means that the system should replicate all journal entries already queued to be sent from the source system to the target system before inactivating the remote journal. No additional journal entries will be queued after a request to perform a controlled inactivate. A controlled inactivate is not possible when a journal is in catch-up, or when it is being synchronously maintained. In both of these cases, the request to perform a controlled inactivate will be implicitly changed by the system to an immediate inactivate request.

1 An immediate inactivate of journal entry replication should be performed. An immediate inactivate means that the system should not continue to replicate any journal entries that are already queued before inactivating the remote journal.

Relational database directory entry. The name of the relational database directory entry that contains the remote location name of the target system.

Remote journal. The name of the remote journal on the target system for which the journal state is being changed, and the library in which it resides. The first 10 characters contain the remote journal name, and the second 10 characters contain the name of the library where the remote journal is located.

Reserved. A reserved space for the purpose of alignment. This field must be initialized to binary 0.

Sending task priority. The priority of the sending task on the source system for asynchronously maintained remote journals. The priority is a value from 1 (highest priority) through 99 (lowest priority), which represents the importance of the task when it competes with other tasks for machine resources. This value represents the relative (not absolute) importance of the task. A special value of 0 indicates that the system will choose a system default for the priority. When the system chooses a priority it is a priority value lower than 1. If this field is not provided, a value of 0 will be assumed.

Starting journal receiver. The journal receiver where the replication of journal entries from the source system to the target system will start.

The possible values follow:

**ATTACHED*
: The replication of journal entries starts with the journal receiver that is currently attached to the remote journal on the target system. The journal entries are replicated from the corresponding journal receiver that is associated with the journal on the source system. The replication starts with the journal entries that follow the last journal entry that currently exists in the attached journal receiver on the target system.

  If the remote journal on the target system does not have an attached journal receiver, the journal receiver that is currently attached to the journal on the source system is created on the target system and attached to the remote journal on the target system. Then journal entries are replicated starting with the first journal entry in the journal receiver that is currently attached to the journal on the source system.

  If the journal on the source system does not have an attached journal receiver, which is only possible in the case of a remote journal that is associated with another remote journal, no journal entries can be replicated and an error is returned.

**SRCSYS*
: The replication of journal entries starts with the journal receiver that is currently attached to the journal on the source system.

  If the corresponding journal receiver exists and is attached to the remote journal on the target system, journal entries are replicated starting with the journal entries that follow the last journal entry that currently exists in the attached journal receiver on the target system. Otherwise, if the corresponding journal receiver exists but is not attached to the remote journal on the target system, no journal entries can be replicated and an error is returned.

  If the corresponding journal receiver does not exist on the target system, the journal receiver is created on the target system and attached to the remote journal on the target system. Then journal entries are replicated starting with the first journal entry in the journal receiver that is currently attached to the journal on the source system.

  If the journal on the source system does not have an attached journal receiver, which is only possible in the case of a remote journal that is associated with another remote journal, no journal entries can be replicated and an error is returned.

Qualified journal receiver name
: The replication of journal entries starts with the specified journal receiver name for the journal on the source system. The first 10 characters contain the journal receiver name, and the second 10 characters contain the name of the library where the journal receiver is located on the source system.

  The special values supported for the library name follow:

  **LIBL* Library list

  **CURLIB*
  Current library

  If the corresponding journal receiver exists and is attached to the remote journal on the target system, journal entries are replicated starting with the journal entries that follow the last journal entry that currently exists in the attached journal receiver on the target system. Otherwise, if the corresponding journal receiver exists but is not attached to the remote journal on the target system, no journal entries can be replicated and an error is returned.

  If the corresponding journal receiver does not exist on the target system, then the journal receiver is created on the target system and attached to the remote journal on the target system. Then journal entries are replicated starting with the first journal entry in the specified journal receiver on the source system.

If the journal on the source system does not have an attached journal receiver, which is only possible in the case of a remote journal that is associated with another remote journal, no journal entries can be replicated and an error is returned.

» **Synchronous sending time-out.** The maximum amount of time in seconds to wait for a response from the remote system when a response is required in a synchronous remote journal environment. If a response is not received within the number of seconds specified, the remote journal environment will be inactivated. If this field is not provided, a value of 0 will be assumed.

The possible values follow:

0	The system chooses the system default of 60 seconds to wait for a response from the remote system.
1-3600	Specify the maximum number of seconds to wait for a response from the remote system.

Validity checking. Specifies whether or not to use communications validity checking. When communications validity checking is enabled, the remote journal environment will provide additional checking to verify that the data which is received by the target system matches the data that was sent from the source system. If the data does not match, the data will not be written to the target system, the remote journal environment will be inactivated, and messages indicating the communications failure will be issued to the journal message queue and QHST. If this field is not provided, a value of 0 will be assumed.

The possible values follow:

0	Communications validity checking will be disabled for this remote journal environment.
1	Communications validity checking will be enabled for this remote journal environment.

Note: Communications validity checking may impact performance. «

Error Messages

Message ID	Error Message Text
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.
CPF3C90 E	Literal value cannot be changed.
CPF69A2 E	State of journal &1 in &2 not changed.
CPF69A3 E	State of journal &1 in &2 not changed.
» CPF69AC E	Synchronous timeout &1 not valid. «
CPF694D E	Unexpected journal receiver &8 found.
CPF694F E	Communications failure.
CPF696A E	Request variable length &1 not valid.
CPF696B E	New journal state &1 not valid.
CPF696C E	Sending task priority &1 not valid.
CPF696D E	Length of receiver variable &1 not valid.
CPF696E E	Type of journal &1 in &2 not valid.
CPF696F E	State of journal &1 in &2 not changed.
CPF697A E	State of journal &1 in &2 not changed.
CPF697B E	State of journal &1 in &2 not changed.
CPF697C E	State of journal &1 in &2 not changed.
CPF697D E	State of journal &1 in &2 not changed.
CPF697E E	State of journal &1 in &2 not changed.

Message ID	Error Message Text
CPF697F E	State of journal &1 in &2 not changed.
CPF6973 E	Systems not compatible.
CPF6974 E	State of journal &1 in &2 not changed.
CPF698A E	State of journal &1 in &2 not changed.
CPF698B E	Unexpected journal receiver attached to &1.
CPF698C E	State of journal &1 in &2 not changed.
CPF698D E	Journal &1 not a remote journal.
CPF698E E	Journal &1 not associated with source journal.
CPF698F E	State of journal &1 in &2 not changed.
CPF6982 E	Relational database directory entry &1 not valid.
CPF699A E	Unexpected journal receiver &8 found.
CPF699C E	Receiver variable parameters not valid.
CPF699D E	Preferred inactivate type &1 not valid.
CPF699E E	State of journal &1 in &2 not changed.
CPF6993 E	State of journal &1 in &2 not changed.
CPF6994 E	State of journal &1 in &2 not changed.
CPF6995 E	Unexpected journal receiver &8 found.
CPF6996 E	Replication of journal entries ended.
CPF6997 E	Unexpected journal receiver &8 found.
CPF6998 E	State of journal &1 in &2 not changed.
CPF6999 E	State of journal &1 in &2 not changed.
CPF70DB E	Remote journal function failed.
CPF70D9 E	Changing journal state not allowed.
CPF701B E	Journal recovery of interrupted operation failed.
CPF8100 E	All CPF81xx messages could be returned. 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.
CPF9810 E	Library &1 not found.
CPF9814 E	Device &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.
CPFBBAC E	The offset to the data port IP address array for node &1 is not valid.
CPFBBAD E	The number of data port IP addresses specified for node &1 is not valid.
TCP1901 E	Internet address &1 is not valid.

API Introduced: V4R2

Top | "Journal and Commit APIs," on page 1 | APIs by category

Clear LU6.2 Partners (QTNCLRLU) API

Required Parameter Group:

1	Partner LU remote network identifier	Input	Char(8)
2	Partner LU location name	Input	Char(8)

Optional Parameter:

3	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Clear LU6.2 Partners (QTNCLRLU) API clears the specified partner logical unit (LU) from the LU6.2 log on this system. In terms of the LU6.2 Peer Protocols, this is known as forcing a cold start with the partner LU during the next connection attempt. This API can be used to eliminate connection problems after the partner LU is moved to a backup system with the same SNA configuration as the original system.

The following informational message is sent to the joblog of the job issuing the API to identify each partner that is cleared:

CPI83DB Partner LU &1.&2 cleared.

Clearing a partner LU will be rejected if there is an active protected conversation between this system and the partner, or if resynchronization to the partner is pending due to a prior communications or system failure. In such cases, the following diagnostic messages will be sent to the joblog to identify any partners that were not cleared, and the API will return error message CPF83EF.

CPD83C3 Partner LU &1.&2 not cleared due to active connection.
CPD83C4 Partner LU &1.&2 not cleared due to pending resynchronization.

Authorities and Locks

Authority
*ALLOBJ special authority is required.

Locks None.

Required Parameters

Partner LU remote network identifier
INPUT; CHAR(8)

The remote network identifier of the partner that is to be cleared.

**ALL* All partners with the specified partner LU location name will be reset. All partners known to this system will be cleared if *ALL is specified for both partner LU remote network identifier and partner LU location name.

Partner LU location name
INPUT; CHAR(8)

The location name of the partner that is to be cleared.

**ALL* All partners with the specified partner LU remote network identifier will be cleared. All partners known to this system will be cleared if *ALL is specified for both partner LU remote network identifier and partner LU location name.

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 caller of the API.

Usage Notes

This API was designed so that it would be easy to use from a CL command line. For example, the following CL command will clear partner APPC.SYSTEM1 from the LU6.2 log:

```
CALL PGM(QTNCLRLU) PARM(APPC SYSTEM1)
```

Error Messages

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF83ED E	&1 API requires &2 special authority.
CPF83EE E	Partner LU &1.&2 is not known to this system.
CPF83EF E	At least one partner LU was not cleared. See previous messages.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R3

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Delete Pointer Handle (QjoDeletePointerHandle) API

Required Parameter:

1	Pointer handle	Input	Binary(4)

Omissible Parameter:

2	Error code	I/O	Char(*)

Service Program: QJOURNAL
Threadsafe: No

The Delete Pointer Handle (QjoDeletePointerHandle) API deletes the specified pointer handle. This pointer handle was generated using the Retrieve Journal Entries (QjoRetrieveJournalEntries) API. See the "Retrieve Journal Entries (QjoRetrieveJournalEntries) API" on page 63 for more information. The deletion of the pointer handle must occur from the same process that called the Retrieve Journal Entries (QjoRetrieveJournalEntries) API, in which the point handle was created. If the handle is not valid because it does not exist or has already been deleted, an error message will be sent.

Authorities and Locks

None

Required Parameter

Pointer handle

INPUT; BINARY(4), UNSIGNED.

The pointer handle to be deleted.

Omissible 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.
CPF3CF1 E	Error code parameter not valid.
CPF6947 E	A pointer handle not deleted.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V4R4

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

End Journal (QjoEndJournal) API

Required Parameter Group:

1	Object entry	Input	Char(*)
2	File ID entry	Input	Char(*)

Omissible Parameter Group:

3	Journal	Input	Char(*)
4	End journal options	Input	Char(*)
5	Error code	I/O	Char(*)

Service Program Name: QJOURNAL
Default Public Authority: *USE
Threadsafe: Yes

The End Journal (QjoEndJournal) API is used to end journaling changes (made to an object or list of objects) to a specific journal. All objects of object type *DIR, *STMF, *SYMLNK, *DTAARA or *DTAQ that currently are being journaled to a specific journal also may have journaling stopped. For objects of type *STMF, *DIR, or *SYMLNK, only objects in the "root" (/), QOpenSys, and user-defined file systems are supported.

Note: For other ways to end journaling, see the following CL commands:

- Integrated File System objects - End Journal (ENDJRN)
- Access Paths - End Journal Access Path (ENDJRNAP)
- Data Areas and Data Queues - End Journal Object (ENDJRNOBJ)
- Physical Files - End Journal Physical File (ENDJRNPF)
- » Libraries - End Journal Library (ENDJRNLIB)«

Restrictions:

1. If a journal name and a list of object names are specified, all objects currently must be journaled to the indicated journal.
2. At least one of parameter object entry or file ID entry must not be NULL.
3. The specified journal must be a local journal.

Authorities and Locks

Journal Authority
*OBJOPR, *OBJMGT

Non-IFS Object Authority (if specified)
*OBJOPR, *READ, *OBJMGT

IFS Object Authority (if specified)
*R, *OBJMGT (also *X if object is a directory and *ALL is specified for the directory subtree key)

Directory Authority (for each directory preceding the last component in the path name)
*X

Journal Lock
*EXCLRD

Non-IFS Object Lock (if specified)
*EXCLRD

IFS Object Lock (if specified)
O_RDONLY | O_SHARE_RDWR

Required Parameters

Object entry
INPUT; CHAR(*)

The path name of the object for which changes are no longer to be journaled. If the object path contains *ALL, all objects supported by this API that currently are being journaled to the indicated journal are to stop having their changes journaled. If the object path is *ALL, then the file ID entry must be NULL.

If this parameter is NULL, the file ID entry parameter must not be NULL.

The object entry must be in the following format.

Object Entry Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number in array
4	4	CHAR(12)	Reserved
Note: These fields repeat for each object path name.			
16	10	BINARY(4)	Length of this path name entry
20	14	CHAR(10)	Include or omit
30	1E	CHAR(2)	Reserved
32	20	CHAR(*)	Object path name

Number in array. The number of objects in the object entry list. The possible values are 1 through 300.

Length of this path name entry. The length of the current path name entry that can be used as the displacement from the start of this path name entry to the next path name entry. The length must be a minimum of 32 bytes and must be a multiple of 16.

Include or omit. Whether the path name is included or omitted from the end journal operation. If the object path is *ALL, then the include or omit paramter is ignored.

**INCLUDE* Objects that match the object name path are to end journaling, unless overridden by an *OMIT specification.

**OMIT* Objects that match the object name path are not to end journaling. This overrides any *INCLUDE specification and is intended to be used to omit a subset of a previously selected path.

Object path name. The object path name for which changes are no longer to be journaled. All relative path names are relative to the current directory at the time of the call to QjoEndJournal.

In the last component of the path name, an asterisk (*) or a question mark (?) can be used to search for patterns of names. The * tells the system to search for names that have any number of characters in the position of the * character. The ? tells the system to search for names that have a single character in the position of the ? character. Symbolic links within the path name will not be followed. If the path name begins with the tilde (~) character, then the path is assumed to be relative to the appropriate home directory.

If a pointer is specified in the object path name, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the path name format, see Path name format.

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

File ID entry
INPUT; CHAR(*)

File identifiers (FID) for which changes are no longer to be journaled.

If the pointer to this parameter is NULL, the object entry parameter must not be NULL.

The structure of this parameter follows.

Object Identifier Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number in array
4	4	CHAR(12)	Reserved
Note: These fields repeat for each file identifier.			
4	4	CHAR(16)	File Identifier

Number in array. The number of objects in the file identifier list. The possible values are 1 through 300.

File identifier. The unique 16-byte file identifier (FID) associated with integrated file system-related objects.

Omissible Parameters

Journal
INPUT; CHAR(*)

The path name of the journal to which changes currently are being journaled. All relative path names are relative to the current directory at the time of the call to QjoEndJournal.

If the journal path name entry contains *OBJ, the path name of the journal is determined by the system from the specified object path name or object file identifier.

If the journal parameter is NULL, *OBJ is assumed.

If a pointer is specified in the path name of the journal, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the journal path name format, see Path name format.

End journal options

INPUT; CHAR(*)

The end journal options, if any, to use for the selection of objects to end journaling changes. If this parameter is not specified, objects will have journaling ended using the default actions described in the field descriptions of the valid keys. See “Keys” on page 39 for the list of valid keys.

This parameter must be specified, but may be a NULL pointer.

You may specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each option must be 16-byte aligned. If not, unpredictable results may occur.

The information must be in the following format.

Journal Options Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number of option records
4	4	CHAR(12)	Reserved
Note: These fields repeat for each option record.			
16	10	BINARY(4)	Length of option record
20	14	BINARY(4)	Key
24	18	BINARY(4)	Length of data
28	1C	CHAR(4)	Reserved
32	20	CHAR(*)	Data

Number of option records. The total number of all option records. If this field is zero, an error will be returned.

Length of option record.

The length of the option record. This length is used to calculate the starting position of the next option record. If you specify a length of option record that is not equal to the key field’s required option record length, an error message is returned.

Key. Specific action for end journal. See “Keys” on page 39 for the list of valid keys.

Length of data.

The length of the option record. This length is used to calculate the ending position of the data for this option.

If you specify a length of data that is not equal to the key field’s required data length, an error message is returned.

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

Data. The data that is used to determine the journal option. All values are validity checked.

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.

Keys

The following table lists the valid keys for the key field area of the journal options record. For detailed descriptions of the keys, see the "Field Descriptions."

Some messages for this API refer to parameters and values for the End Journal (ENDJRN) command. The following table also can be used to locate the key names that correspond to the ENDJRN command parameters.

Key	Input Type	Field	Length of Option Record	Length of Data	ENDJRN Command Parameter
1	CHAR(5)	Directory Subtree	32	5	SUBTREE
2	CHAR(48)	Name Pattern	64	48	PATTERN
» 3	CHAR(1)	Logging Level	32	1	LOGLVL «

Field Descriptions

Directory subtree. Whether the directory subtrees are included in the end journal operation. If this parameter is not specified, the default is *NONE.

This parameter is ignored if the object entry parameter is not specified or if the object is not a directory.

**NONE* Only the objects that match the selection criteria are processed. The objects within selected directories are not processed implicitly.

**ALL* All objects that meet the selection criteria are processed in addition to the entire subtree of each directory that matches the selection criteria. The subtree includes all subdirectories and the objects within those subdirectories.

Include or omit. Whether the name pattern is included or omitted from the operation.

**INCLUDE* Objects that match the object name pattern are processed, unless overridden by an *OMIT specification.

**OMIT* Objects that match the object name pattern are not processed. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Length of this pattern entry. The length of this pattern entry. It is used to calculate the position of the next pattern entry. This field must be set to 32.

» **Logging level.** Specifies the error logging level used. This key is used to determine which messages will be sent. If this key is not specified, the default is 2.

1 *ALL - The API sends all the messages that would be sent with *ERRORS and it will also send the successful completion message for each object.

2 *ERRORS - All diagnostic and escape messages are sent but the API will not send successful completion messages for each object. At the completion of this API, one completion message will be sent. «

Name pattern. The patterns to be used to include or omit objects for processing. The default will be to include all patterns that match.

Additional information about path name patterns is in the Integrated file system topic collection.

Note: This parameter is ignored if the object entry parameter is not specified.

Name Pattern Format

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number in array
8	8	CHAR(12)	Reserved
Note: These fields repeat for each name pattern.			
16	10	BINARY(4)	Length of this pattern entry
20	14	CHAR(10)	Include or omit
30	1E	CHAR(2)	Reserved
32	20	PTR(16)	Pointer to pattern path structure

Number in array. The number of patterns in the pattern list. The possible values are 1 through 20.

Pointer to pattern path structure. A pointer to a path structure.

This pointer must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the pattern path name format, see Path name format.

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

Error Messages

The following messages may be sent from this API:

Message ID	Error Message Text
CPFA0D4 E	File system error occurred.
» CPF0B3F E	The reserved area is not set to binary zeros.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF694A E	Number in array value &1 for key &2 not valid.
CPF694B E	Length &1 of variable record for key &2 not valid.
CPF694C E	Variable length record data for key &1 not valid. «
CPF700B E	&1 of &2 objects ended journaling.
CPF705A E	Operation failed due to remote journal.
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.
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. «

Example

The following example ends journaling a directory object and all objects within that directory subtree. Additionally, it ends journaling on another object identified by its file ID.

Note: By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 180.

```
#include <string.h>
#include <qjournal.h>

void main()
{
    Qjo_Object_Entry_Array_t objectEntryArray;
    Qjo_File_ID_Entry_Array_t fileIDEntryArray;

    struct PathNameStruct
    {
        Qlg_Path_Name_T header;
        char p[50];
    };

    struct PathNameStruct path;
    struct PathNameStruct journalPath;

    char pathName[] = "/CustomerData";

    Qp0lFID_t fileID;

    struct JournalOptionsStruct
    {
        Qjo_Journal_Options_t     opts;
        char spaceForAdditionalOptions[200];
    };

    struct JournalOptionsStruct journalOptions;
    Qjo_Option_t *optionP;

    Qus_EC_t errorCode;


    /* Setup the object's path name structure.                */
    memset(&path name, 0, sizeof(path));
    path.header.CCSID = 37;
    memcpy(path.header.Country_ID,"US",2);
    memcpy(path.header.Language_ID,"ENU",3);
    path.header.Path_Type = QLG_CHAR_SINGLE;
    path.header.Path_Length = strlen(pathName);
    path.header.Path_Name_Delimiter[0] = '/';
    memcpy(path.p, pathName, path.header.Path_Length);

    /* Setup the object entry array.                          */
    memset(&objectEntryArray,0,sizeof(objectEntryArray));
    objectEntryArray.Number_In_Array = 1;

    objectEntryArray.Entry[0].Length_Of_Object_Entry =
      sizeof(objectEntryArray.Entry[0]);
    memcpy(objectEntryArray.Entry[0].Include_Or_Omit,
           QJO_INC_ENT_INCLUDE,
           sizeof(objectEntryArray.Entry[0].Include_Or_Omit));
    objectEntryArray.Entry[0].Path_Name =
      (Qlg_Path_Name_T *)&path;

    /* Get an object's file ID.
       This example is not including the retrieval of the
       file ID for an object.  The user can see the
       Qp0lGetAttr API for information on retrieving an
       object's file ID.  This example will proceed as if the
       fileID variable is set to a valid file ID.          */

    /* Setup the file ID entry array.                         */
    memset(&fileIDEntryArray,0,sizeof(fileIDEntryArray));
    fileIDEntryArray.Number_In_Array = 1;
    memcpy(&fileIDEntryArray.Entry,
           fileID,
```

```
            sizeof(fileIDEntryArray.Entry));

    /* Set the journal options.                                 */
    memset(&journalOptions,0,sizeof(journalOptions));
    journalOptions.opts.Number_Of_Options = 1;

    /* Set the subtree processing images key.                   */
    optionP = (Qjo_Option_t *)&journalOptions.opts.Option[0];

    optionP->Length_Of_Record = QJO_KEY_MINIMUM_RECORD_LENGTH;
    optionP->Key = QJO_KEY_SUBTREE;
    optionP->Length_Of_Data = QJO_KEY_SUBTREE_LENGTH;
    memcpy(optionP->Data,
           QJO_SUBTREE_ALL,
           QJO_KEY_SUBTREE_LENGTH);

    /* Setup the error code structure to cause an exception
       to be sent upon error.                                   */
    memset(&errorCode,0,sizeof(errorCode));
    errorCode.Bytes_Provided = 0;

    QjoEndJournal(&objectEntryArray,
                  &fileIDEntryArray,
                  (Qlg_Path_Name_T *)NULL,
                  (Qjo_Journal_Options_t *)&journalOptions,
                  &errorCode);
}
```

API introduced: V5R1

Top | "Journal and Commit APIs," on page 1 | APIs by category

Materialize Journal Port Attributes (QusMaterializeJournalPortAttr) API

Required Parameter Group:

1	Receiver	I/O	PTR(SPP)
2	Journal port	Input	Open pointer

Service Program Name: QUSMIAPI
Default Public Authority: *USE
Threadsafe: No

The Materialize Journal Port Attributes (QusMaterializeJournalPortAttr) API allows you to retrieve some of the current attributes of a journal.

This API provides the function of the MATJPAT MI instruction on all security levels of the operation system. See the MATJPAT instruction in the i5/OS® Machine Interface topic collection.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
MCHxxxx E	See i5/OS Machine Interface for exact MCH messages that can be signaled.

API introduced: V3R7

Top | "Journal and Commit APIs," on page 1 | APIs by category

Materialize Journal Space Attributes (QusMaterializeJournalSpaceAttr) API

Required Parameter Group:

1	Receiver	I/O	PTR(SPP)
2	Journal space	Input	PTR(SYP)

Service Program Name: QUSMIAPI
Default Public Authority: *USE
Threadsafe: No

The Materialize Journal Space Attributes (QusMaterializeJournalSpaceAttr) API allows you to retrieve some of the current attributes of a journal receiver.

This API provides the function of the MATJSAT MI instruction on all security levels of the operation system. See the MATJSAT instruction in the i5/OS® Machine Interface topic collection.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
MCHxxxx E	See i5/OS Machine Interface for exact MCH messages that can be signaled.

API introduced: V3R7

Top | "Journal and Commit APIs," on page 1 | APIs by category

Remove Commitment Resource (QTNRMVCR) API

Required Parameter Group:

1	Resource handle	Input	Binary(4)
2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Remove Commitment Resource (QTNRMVCR) API removes an API commitment resource that was added to a commitment definition using the Add Commitment Resource (QTNADDCR) API. For more information about adding resources to a commitment definition, see "Add Commitment Resource (QTNADDCR) API" on page 3.

Once a commitment resource is removed, the resource handle that refers to it is no longer valid. You cannot end commitment control for a commitment definition until all API commitment resources have been removed.

If an End Job (ENDJOB) command is entered or you sign off the job, the system automatically ends commitment control for all commitment definitions for the job. Likewise, the system will automatically end an activation-group-level commitment definition for a nondefault activation group that is ending. Any API commitment resources that have not yet been removed from any commitment definition being automatically ended by the system will be implicitly removed by the system during the end job or the activation group end processing. Prior to the system implicitly removing the API commitment resources and automatically ending a commitment definition, an implicit commitment control operation is

performed by the system if pending changes exist for the commitment definition, with the appropriate exit program calls made for any API commitment resources. An implicit commit is performed by the system if the activation group is ending normally. An implicit rollback is performed by the system if the activation group is ending abnormally or the job is ending. For more information about the exit program and information that is passed to it, see "Commitment Control Exit Program" on page 164.

Authorities and Locks

None.

Required Parameter Group

Resource handle
INPUT; BINARY(4)

The resource handle returned by the QTNADDCR API when the API commitment resource was added to the commitment definition.

Error code
I/O; CHAR(*)

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

Restrictions

You are prevented from removing a commitment resource using this API when:

- The resource handle is not valid.
- Commitment control is not active for the program making the request to remove the commitment resource.
- A commitment control operation is currently in progress for the commitment definition that is to have the commitment resource removed from the commitment definition.

In all other instances, the API commitment resource is removed from the commitment definition.

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.
CPF8362 E	Request for commit resource is not valid; reason code &1.
CPF8367 E	Cannot perform commitment control operation.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Remove Remote Journal (QjoRemoveRemoteJournal) API

Required Parameter Group:

1	Qualified journal name	Input	Char(20)
2	Relational database directory entry	Input	Char(18)

Omissible Parameter Group:

3	Request variable	Input	Char(*)
4	Length of request variable	Input	Binary(4)
5	Format name of request variable	Input	Char(8)
6	Error code	I/O	Char(*)

Service Program: QJOURNAL
Default Public Authority: *USE
Threadsafe: No

The Remove Remote Journal (QjoRemoveRemoteJournal) API disassociates a remote journal on the specified target system from the specified journal on the source system. The journal on the source system may be either a local journal or another remote journal.

The remote journal, and any associated journal receivers, are not deleted from the target system by the API processing. No processing is performed on the target system for the API. The remote journal that remains on the target system may later be added back to the remote journal definition for the journal on the source system by using the Add Remote Journal (ADDRMTJRN) command or the Add Remote Journal (QjoAddRemoteJournal) API.

It is the responsibility of the user to delete the remote journal and any associated journal receivers from the target system, if so desired.

Once a remote journal has been removed from a journal, all of the journal receivers that are currently in the journal's receiver directory on the source system will no longer be protected from deletion even if the journal entries have not yet been replicated to the remote journal.

Restrictions

The following restrictions apply:

- The API must be called from the source system for a local or remote journal.
- The remote journal on the specified target system cannot have a journal state of *ACTIVE.

Authorities and Locks

Source Journal Authority
*CHANGE, *OBJMGT

Source Journal Library Authority
*EXECUTE

Service Program Authority
*EXECUTE

Source Journal Lock
*EXCLRD

Required Parameter Group

Qualified journal name
INPUT; CHAR(20)

The name of the journal on the source system from which the remote journal is being removed, and the library where it resides. The journal on the source system may be either a local journal or a remote journal. The first 10 characters contain the journal name, and the second 10 characters contain the name of the library where the journal is located on the source system.

The special values supported for the library name follow:

**LIBL* Library list

**CURLIB* Current library

Relational database directory entry
INPUT; CHAR(18)

The name of the relational database directory entry that contains the remote location name of the target system.

Omissible Parameter Group

Request variable
INPUT; CHAR(*)

The request variable structure that describes the input for the Remove Remote Journal (QjoRemoveRemoteJournal) API.

Length of request variable
INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be set to 20 bytes.

Format name of request variable
INPUT; CHAR(8)

The format RMRJ0100 is the only supported format that is used by this API. See "RMRJ0100 Format" for more information on the RMRJ0100 format.

Error code
I/O; CHAR(*)

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

RMRJ0100 Format

The following table defines the information that may be provided for format RMRJ0100 when you remove a remote journal.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(20)	Qualified remote journal name

Field Descriptions

Qualified remote journal name. The qualified name of the remote journal on the target system. The first 10 characters contain the remote journal name, and the second 10 characters contain the name of the library where the remote journal resides on the target system. If this field is not provided or is blank, the resolved qualified journal name is assumed.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF6981 E	Remote journal &1 in &2 not removed.

Message ID	Error Message Text
CPF6982 E	Relational database directory entry &1 not valid.
CPF6992 E	Remote journal &1 in &2 not removed.
CPF8100 E	All CPF81xx messages could be returned. 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.
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: V4R2

Top | "Journal and Commit APIs," on page 1 | APIs by category

Replay Database Operation (QDBRPLAY) API

Required Parameter Group:

1	Input template	Input	Char(*)
2	Length of input template	Input	Binary(4)
3	Input template format name	Input	Char(8)
4	Journal entry specific data	Input	Char(*)
5	Length of journal entry specific data	Input	Binary(4)
6	Rename exit program scratchpad	Input	Char(*)
7	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Replay Database Operation (QDBRPLAY) API replays a database operation from a single journal entry.

Only database journal entries are supported. Since these journal entries can be quite large, use the Retrieve Journal Entries (QjoRetrieveJournalEntries) API or the Receive Journal Entry (RCVJRNE) command to retrieve the journal entry. If a journal entry is passed to QDBRPLAY that is not supported, the operation will fail. The following journal entries are supported:

Journal Code	Entry Type	Description
D	AC	Add Constraint
F	CB	Change Member
D	CG	Change File
D	CT	Create File
D	DC	Remove Constraint
F	DM	Remove Member
D	DT	Delete File
D	FM	Move File
D	FN	Rename File
D	GC	Change Constraint
D	GO	Change Owner
D	GT	Grant File

Journal Code	Entry Type	Description
F	MC	Add Member
F	MN	Rename Member
F	RM	Reorganize Member
D	RV	Revoke File
D	TC	Add Trigger
D	TD	Remove Trigger
D	TG	Change Trigger
D	TQ	Refresh Table

You can use the QDBRPLAY API with database objects only. DDM files are not supported. File overrides do not affect the specified object names. The API does not run under commitment control even if the original journal entry was performed as part of a commitable transaction. If the specified file does not have the same File Level Identifier or Member Level Identifier as the file for which the journal entry was originally written, a warning will sent to the job log and the operation will continue.

Authorities and Locks

Object Library Authority

*EXECUTE

Note: Additionally, the same authority is necessary as the original operation. For example, if the journal entry is Create File, *ADD is required.

Object Authorities

The same authority is necessary as was required for the original operation. For example, if the journal entry is Delete File, *OBJEXIST is required.

Object Lock

*EXCL or *EXCLRD for *FILE objects.

Note: The same locks are necessary as were required during the original operation. For example, if the journal entry is Delete File, *EXCL is required. If the journal entry is Remove Member, *EXCLRD is required. Because an exclusive lock is required, concurrent applications may not access the file object identified by this API till the API has ended and any concurrent applications that hold a conflicting lock cause the API to fail.

Rename Exit Program Library Authority

*EXECUTE

Rename Exit Program Authority

*EXECUTE

Required Parameter Group

Input template
INPUT;CHAR(*)

A structure that contains the input options used to replay the operation from the journal entry. For the format of this parameter, see “DBRR0100 Format” on page 50.

Length of input template
INPUT; BINARY(4)

A variable that contains the length of the input template. The length must be greater than zero and large enough to contain all the template fields up to and including Disable Triggers. The length must not be larger than 32767.

Input template format name
INPUT; CHAR(8)

The format of the input template being used. The possible value is:

DBRR0100 Basic template

For more information, see "DBRR0100 Format" on page 50.

Journal entry specific data
INPUT;CHAR(*)

The entry specific data from a database journal entry. If the original journal entry contained any additional journal entry specific data that was addressed by a pointer, that data must be moved so that the entry specific data includes all the entry specific data immediately preceding the pointer concatenated with the entry specific data that the pointer addresses. For example, if a teraspace pointer is returned on the QjoRetrieveJournalEntries API immediately after Entry Specific Data A and points to additional Entry Specific Data B:

Entry Specific Data A
Teraspace pointer to additional data B

This must be passed to the API as:

Entry Specific Data A
Additional Entry Specific Data B

The teraspace pointer is in the last 16 bytes of the Entry Specific Data of the retrieved journal entry. The length of the data that the teraspace pointer addresses is equal to the sum of the lengths of the external command length, apply information length, and the remove information length. See member QDBJRNL in QSYSINC/H for more information.

See the Journal management topic collection for information about Entry Specific Data and Additional Entry Specific Data.

Note: The entry specific data for the database operations may be quite large, but is never larger than what will fit in a 16 megabyte space.

Length of journal entry specific data

INPUT;BINARY(4)

The length of the entry specific data from a database journal entry. The length must be greater than zero and must be the length of the actual entry specific data provided when the journal entry was originally written (including any additional journal entry specific data). The length must not be larger than 16 773 120.

Rename exit program scratchpad

INPUT; CHAR(*)

An area which is passed to the rename exit program. The area can contain any information that the caller of the API wishes to pass on to the rename exit program. A rename exit program scratchpad must be passed even if the rename exit program is not specified.

Error code

I/O; CHAR(*)

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

DBRR0100 Format

The following table shows the format of the input template parameter for the DBRR0100 format. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Journal code
1	1	CHAR(2)	Entry type
3	3	CHAR(10)	Rename exit program name
13	D	CHAR(10)	Rename exit program library name
23	17	CHAR(1)	Disable triggers
24	18	CHAR(*)	Reserved

Field Descriptions

Disable triggers. The disable trigger indicator controls whether new triggers that are added as a result of replaying a TC journal entry should be automatically disabled. The disable trigger indicator does not apply to Instead Of triggers.

0 Do not disable new triggers.

1 Disable new triggers.

Entry type. The journal entry type from the journal entry of the operation to replay. See the Journal management topic for more information about Entry type.

Note that a journal entry with an entry type of MN is written to the journal for both a rename member operation and for the internal operations performed as part of a rename file. If a journal entry type of MN that was written to the journal as part of a rename file operation is passed to this API, it will be ignored and no error will be returned.

Journal code. The journal code from the journal entry of the operation to replay. See the Journal management topic for more information on journal codes.

D Database file operation.

F Database file member operation.

Rename exit program name. The name of the rename exit program.

**NONE*

A rename exit program is not provided. The names of any objects referenced during the replay of the operation will be the same as the object names that were referenced when the journal entry was originally written to the journal.

program-name
The name of the program to call which may provide a different name for any object referenced in the journal entry. When a rename exit program is specified, each name referenced during the replay of the operation will be passed to the rename exit program. The names passed to the rename exit program may be short names or long SQL names. The same name may be passed to the exit program more than once if it is referenced in the internal journal entry specific data more than once. If the names are changed by the rename exit program, the names are case sensitive and must conform to any i5/OS® and SQL rules for object names. For example, if the new name of the object should be ″a″, it must be returned from the rename exit program with the quote delimiters and a lower case a. If the new name of the object should be A, it must be returned from the rename exit program in upper case without redundant quote delimiters. The following restrictions apply to referenced objects:

- The library name of an SQL type or an SQL function can be changed. The SQL type name or SQL function name cannot be changed.
- Any references to objects in the body of a program object or trigger are not renamed.
- Any references to objects in the body of an SQL view other than the based on files themselves are not renamed.
- The name of a data dictionary must be the same name as its containing library.
- The library name of a constraint must be the same as the library name of its file.

Two parameters are passed to the rename exit program. The first parameter is the Rename Exit Program Parameter Template. The second parameter is the Rename Exit Program Scratchpad. For more information, see "**Rename Exit Program Parameter**." The exit program may inspect the name passed and leave the name as is, or it may change the name in the Rename Exit Program Parameter to an alternate name.

For example, the journal entry contains the create of an SQL table that contains a primary key constraint. The exit program will be called twice. The first call will pass the table name and library. The second call will pass the constraint name and library. If the name is a reference to another object and is changed to reference an object that does not exist, the replay of the operation will fail.

If the rename exit program returns an exception, the replay operation will fail.

Rename exit program library name. The library that contains the rename exit program, if any. The rename exit program library name must be blank if the value of the rename exit program name is *NONE.

library-name
The library name of the rename exit program.

Reserved. A reserved field. It must contain hexadecimal zeroes.

Rename Exit Program Parameter

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of rename exit program parameter
4	4	BINARY(4)	Length of object name
8	8	BINARY(4)	Length of object library
12	C	BINARY(4)	Object type
16	10	CHAR(258)	Object name
274	112	CHAR(258)	Object library name

Offset		Type	Field
Dec	Hex		
538	21A	CHAR(*)	Reserved

Field Descriptions

Length of rename exit program parameter. The length of the structure passed to the rename exit program. The rename exit program must not modify this value.

Length of object library name. The length of the library name of the referenced object. If the rename exit program modifies this length, it must be greater than zero and not greater than 10. It must reflect the number of characters in the new object library name.

Length of object name. The length of the name of the referenced object. If the rename exit program modifies this length, it must be greater than zero and must reflect the number of bytes in the new object name.

If the value that was passed to the rename exit program is less than or equal to 10 and the rename exit program modifies this length, the new length must also be less than or equal to 10 (a short system name cannot be renamed to a long SQL name). If the value that was passed to the rename exit program is greater than 10 the rename exit program must not modify this length (a long SQL name cannot be renamed to a short name or a long SQL name of a different length).

Object library name. The library name of the referenced object. If the rename exit program modifies the library name, it must be a valid library name.

library-name
The library name of the referenced object.

Object name. The name of the referenced object. If the rename exit program modifies the object name, it must be a valid short object name or a valid long SQL object name.

If the name passed to the exit program is not a long SQL object name, the rename exit program must not rename the object to a long SQL object name. If the name passed to the exit program is a long SQL object name, the rename exit program may rename the long SQL name to another long SQL name, but the length of the new long SQL name must be the same as the long SQL name passed to the rename exit program.

object-name
The name of the referenced object. The name may be either a short (10 character) object name or a long (258 character) SQL object name.

Object type. The type of the referenced object. The following values may be passed to the exit program for the object type:

1 The object attribute is a constraint.
2 The object is an SQL function (*PGM or *SRVPGM).
3 The object is a file (*FILE).
4 The object is a program object (*PGM).
5 The object attribute is a trigger.
6 The object is an SQL type (*SQLUDT).
7 The object is an data dictionary (*DTADCT).
8 The object is a sort sequence or translate table (*TBL).
9 The object is a node group (*NODGRP).
10 The object is a journal (*JRN).

The rename exit program must not modify this value.

While the rename exit program will be called for referenced journals, data dictionaries, and SQL types, these objects themselves cannot cannot be renamed. Furthermore, a library that contains one of these object types cannot be renamed.

Reserved. A reserved field. The rename exit program must not modify this value.

Usage Notes

If a file is created as a result of replaying a Create File (CT) entry:

- If journaling was implicitly started when an SQL table was originally created, the replay operation will also implicitly start journaling to the same journal.
- The File level identifier for the created file will be the same as the File level identifier of the file when it was originally created.
- Any public authority that was specified by the AUT keyword when a file is created via a CL command will be granted when the file is created.

If a member is added as a result of replaying an Add Member operation (MC), the Member level identifier for the added member will be the same as the Member level identifier of the member when it was originally added.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C39 E	Value for reserved field not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3200 E	All CPF32xx messages could be returned. xx is from 01 to FF.
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.
SQL0113 E	Name is not valid.

API introduced: V5R3

Top | Database and File APIs | "Journal and Commit APIs," on page 1 | APIs by category

Replay Journal Entry (QjoReplayJournalEntry) API

Required Parameter Group:

1	Input template	Input	Char(*)
2	Length of input template	Input	Binary(4)
3	Input template format name	Input	Char(8)
4	Journal entry specific data	Input	Char(*)
5	Length of journal entry specific data	Input	Binary(4)

6	Rename exit program scratchpad	Input	Char(*)
7	Error code	I/O	Char(*)

Service Program Name: QJOURNAL
Header File Name: QSYSINC/H.QJOURNAL
Default Public Authority: *USE
Threadsafe: No

The Replay Journal Entry (QjoReplayJournalEntry) API replays a single journal entry.

Only certain journal entries are supported. If a journal entry is passed that is not supported, the operation will fail. The following journal entries are supported:

Journal Code	Entry Type	Description
E	EE	Create data area
Q	QA	Create data queue

Authorities and Locks

Object Library Authority

*EXECUTE

Note: Additionally, the same authority is necessary as the original operation. For example, if the journal entry is Create data area, *ADD is required.

Object Authorities

The same authority is necessary as was required for the original operation.

Object Lock

The same locks are necessary as were required during the original operation.

Rename Exit Program Library Authority

*EXECUTE

Rename Exit Program Authority

*EXECUTE

Required Parameter Group

Input template
INPUT;CHAR(*)

A structure that contains the input options used to replay the operation from the journal entry. For the format of this parameter, see "JORR0100 Format" on page 55.

Length of input template
INPUT; BINARY(4)

A variable that contains the length of the input template. The length must be greater than zero and large enough to contain all the template fields up to and including Rename exit program library name. The length must not be larger than 32767.

Input template format name
INPUT; CHAR(8)

The format of the input template being used. The possible value is:

JORR0100 Basic template

For more information, see "JORR0100 Format."

Journal entry specific data
INPUT;CHAR(*)

The entry specific data from a journal entry.

See the Journal management topic collection for information about entry specific data.

Length of journal entry specific data

INPUT;BINARY(4)

The length of the entry specific data from a journal entry. The length must be greater than zero and must be the length of the actual entry specific data provided when the journal entry was originally written. The length must not be larger than 16 773 120.

Rename exit program scratchpad

INPUT; CHAR(*)

An area which is passed to the rename exit program. The area can contain any information that the caller of the API wishes to pass on to the rename exit program. A rename exit program scratchpad must be passed even if the rename exit program is not specified.

Error code

I/O; CHAR(*)

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

JORR0100 Format

The following table shows the format of the input template parameter for the JORR0100 format. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Journal code
1	1	CHAR(2)	Entry type
3	3	CHAR(10)	Rename exit program name
13	D	CHAR(10)	Rename exit program library name
23	17	CHAR(*)	Reserved

Field Descriptions

Entry type. The journal entry type from the journal entry of the operation to replay. See the Journal management topic collection for more information about entry type.

Journal code. The journal code from the journal entry of the operation to replay. See the Journal management topic collection for more information about journal codes.

E Data area operation.

Q Data queue operation.

Rename exit program name. The name of the rename exit program.

**NONE*
A rename exit program is not provided. The names of any objects referenced during the replay of the operation will be the same as the object names that were referenced when the journal entry was originally written to the journal.

program-name
The name of the program to call which may provide a different name for any object referenced in the journal entry. When a rename exit program is specified, each name referenced during the replay of the operation will be passed to the rename exit program. The same name may be passed to the exit program more than once if it is referenced in the internal journal entry specific data more than once. If the names are changed by the rename exit program, the names are case sensitive and must conform to any i5/OS® rules for object names.

Two parameters are passed to the rename exit program. The first parameter is the Rename Exit Program Parameter Template. The second parameter is the Rename Exit Program Scratchpad. For more information, see "**Rename Exit Program Parameter**." The exit program may inspect the name passed and leave the name as is, or it may change the name in the Rename Exit Program Parameter to an alternate name.

If the rename exit program returns an exception, the replay operation will fail.

Rename exit program library name. The library that contains the rename exit program, if any. The rename exit program library name must be blank if the value of the rename exit program name is *NONE.

library-name
The library name of the rename exit program.

Reserved. A reserved field. It must contain hexadecimal zeroes.

Rename Exit Program Parameter

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of rename exit program parameter
4	4	BINARY(4)	Length of object name
8	8	BINARY(4)	Length of object library
12	C	BINARY(4)	Object type
16	10	CHAR(258)	Object name
274	112	CHAR(258)	Object library name
538	21A	CHAR(*)	Reserved

Field Descriptions

Length of object library name. The length of the library name of the referenced object. If the rename exit program modifies this length, it must be greater than zero and not greater than 10. It must reflect the number of characters in the new object library name.

Length of object name. The length of the name of the referenced object. If the rename exit program modifies this length, it must be greater than zero and not greater than 10, and must reflect the number of bytes in the new object name.

Length of rename exit program parameter. The length of the structure passed to the rename exit program. The rename exit program must not modify this value.

Object library name. The library name of the referenced object. If the rename exit program modifies the library name, it must be a valid library name.

library-name
The library name of the referenced object.

Object name. The name of the referenced object. If the rename exit program modifies the object name, it must be a valid object name.

object-name
The name of the referenced object. The name must be a 10 character object name.

Object type. The type of the referenced object. The following values may be passed to the exit program for the object type:

101 The object is a data area (*DTAARA).
102 The object is a data queue (*DTAQ).

The rename exit program must not modify this value.

Reserved. A reserved field. The rename exit program must not modify this value.

Usage Notes

If a data area or data queue is created as a result of replaying a Create data area (E EE) or Create data queue (Q QA) entry:

- If the object was originally created on a V6R1M0 or later system and journaling was automatically started when the object was created, the replay operation will also automatically start journaling to the same journal. Otherwise the object will not automatically start journaling.
- If the object was originally created on a V6R1M0 or later system and journaling was automatically started when the object was created, the journal identifier for the created object will be the same as the journal identifier of the object when it was originally created. Otherwise the journal identifier will not be preserved.
- The resulting authority on the original create will be granted when the object is created during the replay processing.

Error Messages

Message ID	Error Message Text
CPF019D E	Value &1 not valid as sytem name.
CPF1000 E	All CPF10xx messages could be returned. xx is from 01 to FF.
CPF2100 E	All CPF21xx messages could be returned. xx is from 01 to FF.
CPF2283 E	Authorization list &1 does not exist.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C39 E	Value for reserved field not valid.
CPF3C3C E	Value for parameter &1 not valid.
CPF3CF1 E	Error code parameter not valid.
CPF6565 E	User profile storage limit exceeded.
CPF70C2 E	Value returned from the rename exit program is not valid.
CPF9500 E	All CPF95xx messages could be signaled. xx is from 01 to FF.
CPF9800 E	All CPF98xx messages could be signaled. xx is from 01 to FF.

Retrieve Commitment Information (QTNRCMTI) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Error code	I/0	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Commitment Information (QTNRCMTI) API allows you to determine if commitment control is active within the activation group for the program performing the retrieve request. Other information that can be retrieved includes:

- Default lock level and default journal name and library
- Commitment definition scope
- Activation group commitment definition status
- Commitment options
- Logical unit of work identifier

Information about commitment definitions that are started by the system for system use only is not available through this API.

Authorities and Locks

None.

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

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

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable. The length must be at least 8 bytes. If the variable is not long enough to hold the information, the data is truncated. If the length is larger than the size of the receiver variable, the results are not predictable.

Format name
INPUT; CHAR(8)

The format name CMTI0100 is the only valid format name used by this API. For more information, see "CMTI0100 Format" on page 59.

Error code
I/O; CHAR(*)

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

CMTI0100 Format

The structure of the information returned is determined by the specified format name. For detailed descriptions of the fields, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(1)	Commitment definition status
9	9	CHAR(10)	Commitment definition default lock level
19	13	CHAR(1)	Commitment definition scope
20	14	CHAR(1)	Commitment definition status for the activation group
21	15	CHAR(1)	Wait for outcome commitment option
22	16	CHAR(1)	Action if problems commitment option
23	17	CHAR(1)	Vote read-only permitted commitment option
24	18	CHAR(1)	Action if ENDJOB commitment option
25	19	CHAR(10)	Default journal name
35	23	CHAR(10)	Default journal library name
45	2D	CHAR(39)	Logical unit of work identifier
84	54	CHAR(1)	Last agent permitted commitment option
85	55	CHAR(1)	OK to leave out commitment option
86	56	CHAR(1)	Accept vote reliable commitment option

Field Descriptions

For more detailed descriptions of these fields and their values, see "Commitment Options Format" on page 18 in the Change Commitment Options (QTNCHGCO) API.

Accept vote reliable commitment option. The current value of this commitment option.

The possible values are:

Y The system accepts the vote reliable indicator if it is returned from its agents during the prepare wave of a commit operation. Control is returned to the application before the committed wave is completed for agents that send the vote reliable indicator if this system accepts it. If heuristic damage is encountered during the committed wave at any agents that voted reliable, it is not reported to the application.

N The system does not accept the vote reliable indicator. Control is returned to the application after the committed wave is completed for all agents. If heuristic damage is encountered during the committed wave, it is reported to the application as an escape message.

Action if ENDJOB commitment option. The current value of this commitment option.

The possible values are:

W The system waits to allow the normal processing of the logical unit of work to complete.

R Local resources whose status is in doubt are rolled back in the event of an ENDJOB.

C Local resources whose status is in doubt are committed in the event of an ENDJOB.

Action if problems commitment option. The current value of this commitment option.

The possible values are:

R If a problem is detected and the local resources have not been committed or rolled back, the local resources will be rolled back.
C If a problem is detected and the local resources have not been committed or rolled back, the local resources will be committed.

Bytes available. The length in bytes of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length in bytes of all data actually returned.

Commitment-definition default lock level. The default lock level at the start of commitment control for the commitment definition. This is the level of record locking that applies to records in all commitment resources under commitment control for the commitment definition. The level of record locking is this value unless it is overridden when the file is opened. You cannot override this value; however, the system can override for files opened for system functions.

The possible values are:

blank Blanks are returned when I is returned for commitment definition status.
**ALL* Changed and retrieved records are protected from changes by other jobs running at the same time.
**CHG* Changed records are protected from changes by other jobs running at the same time.
**CS* Changed and retrieved records are protected from changes by other jobs running at the same time. Retrieved records are protected until they are released or until a different record is retrieved.

Commitment definition scope. The scope for the commitment definition currently active within the activation group for the program performing the retrieve request.

The possible values are:

A Activation group level
J Job level

Commitment definition status. The overall status of the commitment definition currently active for the activation group for the program performing the retrieve request. The scope for this commitment definition is returned in the commitment definition scope field.

The possible values are:

I Commitment control is not active at either the activation group level or the job level for the program making the retrieve request.
A The commitment definition is active within the activation group for the program performing the retrieve request. No local, remote, or API commitment resource is associated with the commitment definition.
L The commitment definition is active on the local system within the activation group for the program performing the retrieve request.

An *L* is returned if one or more of the following resources are under commitment control.

- Local, open database files
- Local, closed database files with pending changes
- Resources with object-level changes
- Local relational database resources
- API commitment resources

R The commitment definition is active on one or more remote systems within the activation group for the program performing the retrieve request.
An *R* is returned if one or more of the following resources are under commitment control.

- Remote DDM resources
- Remote distributed relational database resources
- Protected conversations

B The commitment definition is active on both the local and one or more remote systems.
If both *L* and *R* could be returned, a *B* will be returned.

Commitment definition status for the activation group. The status of the commitment definition currently active for the activation group for the program performing the retrieve request, but specifically regarding any commitment resource changes made by programs running within the activation group.

The possible values are:

I Commitment control is not active at the activation group level or the job level for the program making the retrieve request. The overall status for the job-level commitment definition is also not active.

A The commitment definition is currently being used by one or more programs running within this activation group. Local, remote or API commitment resources may have been placed under commitment control to the commitment definition by a program running within the activation group. The overall status for the commitment definition is either active (A), active on the local system (L), active on a remote system (R), or active on both the local and one or more remote systems (B).

N The job-level commitment definition is active but is not in use by any program running within this activation group. The overall status for the commitment definition is either active (A), active on the local system (L), active on a remote system (R), or active on both the local and one or more remote systems (B). It is possible to start the activation-group-level commitment definition for programs running within this activation group.

Default journal name. The journal name specified for the default journal when this commitment definition was started. If no default journal was specified, blanks are returned.

Default journal library name. The journal library name specified for the default journal when this commitment definition was started. If no default journal was specified, blanks are returned.

Last agent permitted commitment option. The current value of this commitment option.

The possible values are:

S The system is allowed to select a last agent.

N The system is not allowed to select a last agent.

Logical unit of work identifier. The identifier for the logical unit of work currently associated with this commitment definition.

Table 1. Logical Unit of Work Identifier Format

Field	Type	Description
Network ID	CHAR(0-8)	Network identifier

Field	Type	Description
Separator	CHAR(1)	The separator character ".″
Local location name	CHAR(0-8)	The name of the local location
Separator	CHAR(3)	The separator characters ".X′"
Instance number	CHAR(12)	The hex value of the instance number converted to decimal
Separator	CHAR(2)	The separator characters "′."
Sequence number	CHAR(5)	The hex value of the sequence number converted to decimal

OK to leave out commitment option. The current value of this commitment option.

The possible values are:

Y If this location does not initiate the commit operation, it may be left out of subsequent logical units of work. Control is not returned to the application until a data flow is received from the initiator.

N This location may not be left out of subsequent logical units of work. Control is returned to the application immediately after the commit operation completes.

Vote read-only permitted commitment option. The current value of this commitment option.

The possible values are:

N This location is not allowed to vote read-only in response to a prepare request from a remote location.

Y This location is allowed to vote read-only in response to a prepare request from a remote location. Control will not be returned to the application until data is sent from the remote location that sent the prepare request.

Wait for outcome commitment option. The current value of this commitment option.

The possible values are:

Y If a remote resource failure occurs during a commit or rollback operation, the system will not return from the operation until resynchronization is complete.

L Value L has the same effect as value Y when this location is the initiator of the commit or rollback operation. When this location is not the initiator and the initiator supports the presumed abort protocol, the Wait for outcome value is inherited from the initiator. When this location is not the initiator and the initiator does not support the presumed abort protocol, value L has the same effect as value Y.

N If a remote resource failure occurs during a commit or rollback operation, the system will attempt to resynchronize with the failed resource once. If the one attempt fails, resynchronization will then be done in a system job and control is returned to the application.

U Value U has the same effect as value N when this location is the initiator of the commit or rollback operation. When this location is not the initiator and the initiator supports the presumed abort protocol, the Wait for outcome value is inherited from the initiator. When this location is not the initiator and the initiator does not support the presumed abort protocol, value U has the same effect as value N.

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.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.

Message ID	Error Message Text
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

Top | "Journal and Commit APIs," on page 1 | APIs by category

Retrieve Journal Entries (QjoRetrieveJournalEntries) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Qualified journal name	Input	Char(20)
4	Format name	Input	Char(8)

Omissible Parameter Group:

5	Journal entries to retrieve	Input	Char(*)
6	Error code	I/O	Char(*)

Service Program Name: QJOURNAL
Header File Name: QSYSINC/H.QJOURNAL
Default Public Authority: *USE
Threadsafe: No

The Retrieve Journal Entries (QjoRetrieveJournalEntries) API provides access to journal entries. The journal entry information available is similar to what is provided by using the Display Journal (DSPJRN), Receive Journal Entry (RCVJRNE), and Retrieve Journal Entry (RTVJRNE) CL commands. Additionally, journal entry data that cannot be retrieved through these CL interfaces because of length or structure is available through this API as pointers to the additional data.

For more information about journaling and the various types of journal entries that are available for retrieval, see the Journal management topic collection.

Note: Under certain conditions, even if an error message is returned to this API, a partial list of journal entries may be retrieved into the receiver variable. If you receive error messages CPF3CF1, CPF3C90, CPF6948, CPF6949 or CPF9872, then the receiver variable has not yet been modified. For all other error messages, a non-zero value for bytes returned indicates journal entry information may be available and the number of entries retrieved field will reflect how many entries are being returned prior to receiving the error message.

Restrictions

- If the sequence number is reset in the range of the receivers specified, the first occurrence of starting sequence number or ending sequence number is used if these key fields are specified.
- The job, program, and user profile keys cannot be used to specify selection criteria if one or more journal receivers in the specified receiver range was attached to a journal that had a RCVSIZOPT or FIXLENDTA option specified that omitted the collection of that data.
- The file, object, object path, object file identifier, directory subtree, name pattern, » object journal identifier «, journal code, entry type, job, program, user profile, commit cycle identifier, and dependent entries keys can be used to specify a subset of all available entries within a range of journal entries.
  - If no values are specified using these keys, all available journal entries are retrieved.

  - If more than one of these keys are specified, then a journal entry must satisfy all of the values specified on these keys, except when ignore file selection (*IGNFILSLT) or ignore object selection (*IGNOBJSLT) is specified on the journal code key.
  - If a journal code is specified on the journal code key and *IGNFILSLT is the second element of that journal code, then journal entries with the specified journal code are selected if they satisfy all selection criteria except what is specified on the file key.
  - If a journal code is specified on the journal code key and *IGNOBJSLT is the second element of that journal code, then journal entries with the specified journal code are selected if they satisfy all selection criteria except what is specified on the object, object path, object file identifier, directory subtree, name pattern, and object journal identifier keys.
- If more than the maximum number of objects is identified (32767 objects), an error occurs and no entries are converted for output. This restriction is ignored if *ALLFILE is specified or no objects are specified. All journal entries are retrieved, regardless of which objects, if any, the entries are associated with.

Authorities and Locks

Journal Authority
*USE

Journal Library Authority
*EXECUTE

Journal Receivers Authority
*USE

Journal Receivers Library's Authority
*EXECUTE

Non-Integrated File System Object Authority (if specified on Key 16 (file) or Key 17 (object))
*USE

Non-Integrated File System Object Library Authority
*EXECUTE

Integrated File System Object Authority (if specified on Key 18 (object path) or Key 19 (object file identifier))
*R (also *X if object is a directory and *ALL is specified for the directory subtree key)

Directory Authority (for each directory preceding the last component in the path name)
*X

Journal Lock
*SHRRD

Journal Receiver Lock
*SHRRD

Non-Integrated File System Object Lock (if specified)
*SHRRD

Integrated File System Object Lock (if specified)
O_RDONLY | O_SHARE_RDWR

*OBJEXIST is also required for the journal authority if any of the the following are true:

- *ALLFILE has been specified for the file key or no objects were specified
- Specified object does not exist on the system
- Specified object has never been journaled
- *IGNFILSLT or *IGNOBJSLT is specified for the journal code selection value for any selected journal codes

- The journal is a remote journal
- Key 23 (object journal identifier) is specified

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The receiver variable that receives the entries requested. You can specify the size of the area smaller than the format requested as long as you specify the length of receiver variable parameter correctly. As a result, the API returns only the data the area can hold. Only complete journal entries will be returned.

Note: This receiver variable must be aligned on a 16-byte boundary since the journal entry specific data could include actual pointers.

If the receiver variable was not large enough to hold the retrieved journal entries, the API can be called again, specifying the same selection criteria and specifying a starting sequence number one greater than the last sequence number returned.

Length of receiver variable
INPUT; BINARY(4)

The length of 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. If the length of receiver variable parameter specified is smaller than the journal entry to be returned, no journal entry will be returned and no error will be signalled. The minimum length is 13 bytes.

Qualified journal name
INPUT; CHAR(20)

The name of the journal and its library from which the journal entries are to be retrieved. The first 10 characters contain the journal name. The second 10 characters contain the library name. The special values supported for the library name follow:

**LIBL*	Library list
**CURLIB*	Current library

Format name
INPUT; CHAR(8)

The formats RJNE0100 and RJNE0200 are the only supported formats that are used by this API. For more information, see the "RJNE0100 Format" on page 81 and the "RJNE0200 Format" on page 83.

Omissible Parameter Group

Journal entries to retrieve
INPUT; CHAR(*)

The selection criteria, if any, to use for the journal entries to be retrieved from the journal. If this parameter is not specified, all journal entries in the currently attached journal receiver that fit in the length of receiver variable parameter will be retrieved. Only complete journal entries will be returned. 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. If this field is zero, no variable length records are processed, and no key information will be retrieved.

Variable length records
CHAR(*)

The types of entries that should be retrieved. For the specific format of the variable length record, see "Format for Variable Length Record."

Note: This receiver variable must be aligned on a 16-byte boundary since the journal entry specific data could include actual pointers.

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.

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)	Length of variable length record
4	4	BINARY(4)	Key
8	8	BINARY(4)	Length of data
12	C	CHAR(*)	Data

If you specify a length of data that is longer than the key field's required data length, the data will be truncated at the right. No error message will be returned.

If you specify a length of data that is shorter than the key field's required data length, an error message will be returned.

You may 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. If any keys are specified which include pointers, e.g. Object Path or Name Pattern, it is recommended that the length of each variable length record be a mulitple of 16 bytes. If not, errors may occur.

Field Descriptions

Data. The data that is used to determine how the journal entries should be retrieved. All values are validity checked.

Key. Identifies specific entries to be retrieved from the journal. See "Keys" on page 67 for the list of valid keys.

Length of data. The length of the key information.

Length of variable length record. The length of the variable length record. This field is used to get the addressability of the next variable length record.

Keys

The following table lists the valid keys for the key field area of the variable length record. For detailed descriptions of the keys, see the "Field Descriptions."

Some messages for this API refer to parameters and values of the Receive Journal Entry (RCVJRNE) command. This table also can be used to locate the key names that correspond to the RCVJRNE command parameters.

Key	Input Type	Field	RCVJRNE Command Parameter
1	CHAR(40)	Range of journal receivers	RCVRNG
2	CHAR(20)	Starting sequence number	FROMENT
3	CHAR(26)	Starting time stamp	FROMTIME
4	CHAR(20)	Ending sequence number	TOENT
5	CHAR(26)	Ending time stamp	TOTIME
6	BINARY(4)	Number of entries	NBRENT
7	CHAR(*)	Journal codes	JRNCDE
8	CHAR(*)	Journal entry types	ENTTYP
9	CHAR(26)	Job	JOB
10	CHAR(10)	Program	PGM
11	CHAR(10)	User profile	USRPRF
12	CHAR(20)	Commit cycle identifier	CMTCYCID
13	CHAR(10)	Dependent entries	DEPENT
14	CHAR(10)	Include entries	INCENT
15	CHAR(10)	Null value indicators length	NULLINDLEN
16	CHAR(*)	File	FILE
17	CHAR(*)	Object	OBJ
18	CHAR(*)	Object path	OBJPATH
19	CHAR(*)	Object file identifier	OBJFID
20	CHAR(5)	Directory subtree	SUBTREE
21	CHAR(*)	Name pattern	PATTERN
22	CHAR(10)	Format Minimized Data	FMTMINDTA
» 23	CHAR(*)	Object journal identifier	OBJJID «

Field Descriptions

Commit cycle identifier. The commit cycle identifier of the specific journal that participated in a logical unit of work for which the journal entries are to be retrieved. This Char(20) field is treated as Zoned(20,0) except when the special value *ALL is specified. The default is *ALL. The possible values are:

**ALL* The journal entries for all commit cycle identifiers are to be retrieved.
commit cycle identifier The identifier for the commit cycle whose journaled changes are to be retrieved.

Dependent entries Whether the journal entries to be retrieved include the journal entries recording actions

- that occur as a result of a trigger program.

- on records that are part of a referential constraint.
- that will be ignored during an apply journaled changes (APYJRNCHG) or remove journaled changes (RMVJRNCHG) operation.

The default is *ALL. The possible values are:

*ALL	The journal entries relating to trigger programs, referential constraints, and the entries that will be ignored by an apply journaled changes or remove journaled changes operation are retrieved.
*NONE	The journal entries relating to trigger programs, referential constraints, and the entries that will be ignored by an apply journaled changes or remove journaled changes operation are not retrieved.

Directory subtree. Whether the directory subtrees are included in the retrieve operation. The default is *NONE.

Note: This key is ignored if Key 18 (object path) is not specified or if the object is not a directory.

*NONE	Only the objects that match the selection criteria are processed. The objects within selected directories are not processed implicitly.
*ALL	All objects that meet the selection criteria are processed in addition to the entire subtree of each directory that matches the selection criteria. The subtree includes all subdirectories and the objects within those subdirectories.

Ending sequence number. The last journal entry considered for retrieval. This Char(20) field is treated as Zoned(20,0) except when the special value *LAST is specified. The default is *LAST. The possible values are:

*LAST	The last journal entry in the specified journal receiver range is the last entry considered for retrieval.
sequence number	The sequence number of the journal entry that is the last entry considered for retrieval.

Note: If this key is specified, Key 5 (ending time stamp) cannot also be specified.

Ending time stamp. The time stamp of the last journal entry considered for retrieval. This Char(26) field is in the format YYYY-MM-DD-HH.MM.SS.UUUUUU where

YYYY	Year
MM	Month
DD	Day
HH	Hours
MM	Minutes
SS	Seconds
UUUUUU	Microseconds

Notes:

1. If this key is specified, Key 4 (ending sequence number) cannot also be specified.
2. If the system value QLEAPADJ (Leap year adjustment) is zero, then the result returned will be in 1 microsecond granularity. If the system value QLEAPADJ is greater than zero, then the result returned will be in 8 microsecond granularity.

File. A list of files and members for which journal entries are to be retrieved. For the format of this field, see "File Format" on page 74. If *ALLFILE is specified for the file name, the list cannot contain other entries.

To determine which journal entries are to be retrieved, based on the specified file member name, the following is done:

- If the journal is a local journal, and if the specified file member currently exists on the system, the journal identifier is determined from the specified file member. All journal entries in the specified receiver range for that journal identifier are retrieved.
- If the journal is a remote journal, or if the specified file member does not currently exist on the system, the specified receiver range is searched to determine all possible journal identifiers that are associated with the specified file member. All journal entries in the specified receiver range for those journal identifiers are retrieved. Specify the library name or *CURLIB to have entries returned for a file.

  There may be more than one journal identifier associated with a specified object within the specified receiver range. This can happen when a journaled object is deleted, and then a new object is created with the same name and journaled to the same journal.

Notes:

1. The journal identifier is the unique identifier associated with the object when journaling is started for that object. The journal identifier stays constant, even if the object is renamed, moved or restored. See the Journal management topic collection for more information.
2. When specifying a database file on this key, journal entries with the following journal code values are retrieved only if they satisfy the values specified on the other keys:
   - Journal code D (database file-level information entries).
   - Journal code F (file member-level information entries).
   - Journal code R (record-level information entries).
   - Journal code U (user-generated entries).
   - Other journal codes if *IGNFILSLT is the second element of the journal code key. If *ALLSLT is the second element of the journal code key, no journal entries with that code are retrieved.
3. Either Key 16 (file) may be specified, or one or more of the object keys, Key 17 (object), Key 18 (object path), Key 19 (object file identifier),

   or Key 23 (object journal identifier)

   may be specified, but not both.

Format minimized data. Specifies whether entry specific data which has been minimized on field boundaries will be returned in a readable format. The default is *NO. The possible values are:

*NO	The journal entries which have entry specific data that has been minimized on field boundaries will not be returned in a readable format. Therefore, the entry specific data may not be viewable.
*YES	The journal entries which have entry specific data that has been minimized on field boundaries will be returned in a readable format. Therefore, the entry specific data is viewable and may be used for auditing purposes. The fields that were changed are accurately reflected. The fields that were not changed and were not recorded return default data and are indicated by a value of '9' in the null value indicators field.

Include entries. Whether only the confirmed or both the confirmed and unconfirmed journal entries are retrieved. This key only applies when retrieving journal entries from a remote journal. The default is *CONFIRMED.

Confirmed entries are those journal entries that have been sent to this remote journal, and the state of the input/output (I/O) to auxiliary storage for the same journal entries on the local journal is known.

Unconfirmed entries can occur for two reasons. First, the journal entries have been sent to the remote journal, but the state of the input/output (I/O) to auxiliary storage for the same journal entries on the local journal is not yet known. If the connection to the source system is lost, these entries will be deleted from the remote system and will never become confirmed. This situation only occurs if synchronous delivery mode is being used for the remote journal. Secondly, unconfirmed entries may exist because the object name information for those journal entries is not yet known to the remote journal. Even if the connection to the source system is lost, these entries will eventually become confirmed. This situation can occur for either synchronous or asynchronous delivery mode to a remote journal.

The possible values are:

**CONFIRMED*	Only those journal entries that have been confirmed are retrieved.
**ALL*	All confirmed and unconfirmed journal entries are retrieved.

Job. Whether the journal entries being retrieved are limited to the journal entries for a specified job. Only journal entries for the specified job are considered for retrieval. The default is *ALL. The possible values are:

**ALL*	The retrieval is not limited to entries for a specific job.
job	The retrieval is limited to entries for a specific job where the first 10 characters are the job name, the second 10 characters are the user name, and the last 6 characters are the job number.

Journal codes. A list of journal codes for which entries are to be retrieved. For the format of this field, see "Journal Code Format" on page 75. If *ALL or *CTL is specified for the journal code value, the list cannot contain other entries and the journal code selection field must be blank. The default is *ALL for the journal code value.

Journal entry types. A list of journal entry types for which entries are to be retrieved. For the format of this field, see "Journal Entry Type Format" on page 76. If *ALL or *RCD is specified for the journal entry type, the list cannot contain other entries. The default is *ALL for the journal entry type.

Name pattern. The patterns to be used to include or omit objects for which journal entries are to be retrieved. The default will be to include all patterns that match. For the format of this field, see "Name Pattern Format" on page 76.

Notes:

1. This key is ignored if Key 18 (object path) is not specified.
2. The sum of the lengths of all the variable length records that precede this key must be a multiple of 16 bytes. If not, errors will occur. For ease of implementation, it is recommended that the length of each variable length record be a mulitple of 16 bytes.

Null value indicators length. The length, in bytes, used for the null value indicators portion of the journal entry retrieved by the user. This Char(10) field is treated as Zoned(10,0) except when the special value *VARLEN is specified. The default is *VARLEN. The possible values are:

**VARLEN*	The null value indicators field is a variable-length field. The received journal entry has the format shown in This journal entry's null value indicators if Null Value Indicators (*VARLEN) specified (page 83). All possible null value indicators will be retrieved.
field length	The null value indicators field is a fixed-length field of the specified field length. Valid values range from 1 to 8000 characters. The format of the retrieved journal entry is shown in This journal entry's null value indicators if Null Value Indicators (field length) specified (page 83).

If the journal entry being retrieved has fewer null value indicators than the length specified, the trailing bytes are set to 'F0'X. Conversely, if a journal entry retrieved has more null value indicators than the specified field length and truncation will result in the loss of a significant null value indicator (either a 'F1'X or a 'F9'X), the request is ended.

Number of entries. The maximum number of journal entries that are requested to be retrieved. Less than this maximum could be retrieved if fewer entries meet all the other selection criteria or if there is not enough space for all the requested entries.

Object. A list of objects for which journal entries are to be retrieved. For the format of this field, see "Object Format" on page 77. Only objects of type *DTAARA, *DTAQ, *FILE, » and *LIB « are supported.

To determine which journal entries are to be retrieved, based on the specified object name, the following is done:

- If the journal is a local journal, and if the specified object currently exists on the system, the journal identifier is determined from the specified object. All journal entries in the specified receiver range for that journal identifier are retrieved.
- If the journal is a remote journal, or if the specified object does not currently exist on the system, the specified receiver range is searched to determine all possible journal identifiers that are associated with the specified object. All journal entries in the specified receiver range for those journal identifiers are retrieved. Specify the library name or *CURLIB to have entries returned for an object.

  There may be more than one journal identifier associated with a specified object within the specified receiver range. This can happen when a journaled object is deleted, and then a new object is created with the same name and journaled to the same journal.

Notes:

1. The journal identifier is the unique identifier associated with the object when journaling is started for that object. The journal identifier stays constant, even if the object is renamed, moved or restored. See the Journal management topic collection for more information.
2. When specifying an object on this key, journal entries with the following journal code values are retrieved only if they satisfy the values specified on the other keys:
   - Journal code D (database file-level information entries).
   - Journal code E (data area information entries).
   - Journal code F (file member-level information entries).
   - Journal code Q (data queue information entries).
   - Journal code R (record-level information entries).
   - Journal code U (user-generated entries).
   - » Journal code Y (library information entries).«
   - Other journal codes if *IGNOBJSLT is the second element of the journal code key. If *ALLSLT is the second element of the journal code key, no journal entries with that code are retrieved.
3. Either Key 16 (file) may be specified, or one or more of the object keys, Key 17 (object), Key 18 (object path), Key 19 (object file identifier), » or Key 23 (object journal identifier)« may be specified, but not both.

Object file identifier. A list of file identifiers (FIDs) for which journal entries are to be retrieved. FIDs are unique identifiers associated with integrated file system objects. Only objects whose FID identifies an object of type *STMF, *DIR, or *SYMLNK that is in the "root" (/), QOpenSys, and user-defined file systems are supported. All other objects are ignored. For the format of this field, see "Object File Identifier Format" on page 78.

To determine which journal entries are to be retrieved, based on the specified file identifier, the following is done:

- If the journal is a local journal, and if the specified object currently exists on the system, the journal identifier is determined from the specified object. All journal entries in the specified receiver range for that journal identifier are retrieved.
- If the journal is a remote journal, or if the specified object does not currently exist on the system, the specified receiver range is searched to determine all possible journal identifiers that are associated with the specified object. All journal entries in the specified receiver range for those journal identifiers are retrieved.

Notes:

1. The journal identifier is the unique identifier associated with the object when journaling is started for that object. The journal identifier stays constant, even if the object is renamed, moved or restored. See the Journal management topic collection for more information.
2. When specifying an object on this key, journal entries with the following journal code values are retrieved only if they satisfy the values specified on the other keys:
   - Journal code B (integrated file system information entries).
   - Journal code U (user-generated entries).
   - Other journal codes if *IGNOBJSLT is the second element of the journal code key. If *ALLSLT is the second element of the journal code key, no journal entries with that code are retrieved.
3. Either Key 16 (file) may be specified, or one or more of the object keys, Key 17 (object), Key 18 (object path), Key 19 (object file identifier), » or Key 23 (object journal identifier) « may be specified, but not both.

» **Object journal identifier.** A list of journal identifiers for which journal entries are to be retrieved. For the format of this field, see Object Journal Identifier Format (page "Object Journal Identifier Format" on page 79).

Notes:

1. The journal identifier is the unique identifier associated with the object when journaling is started for that object. The journal identifier stays constant, even if the object is renamed, moved or restored. See the Journal management topic collection for more information.
2. When specifying a journal identifier on this key, journal entries with the following journal code values are retrieved only if they satisfy the values specified on the other keys:
   - Journal code B (integrated file system information entries).
   - Journal code D (database file-level information entries).
   - Journal code E (data area information entries).
   - Journal code F (file member-level information entries).
   - Journal code J (journal receiver information entries).
   - Journal code R (record-level information entries).
   - Journal code Q (data queue information entries).
   - Journal code U (user-generated entries).
   - » Journal code Y (library information entries). «
   - Other journal codes if *IGNOBJSLT is the second element of the journal code key. If *ALLSLT is the second element of the journal code key, no journal entries with that code are retrieved.
3. Either Key 16 (file) may be specified, or one or more of the object keys, Key 17 (object), Key 18 (object path), Key 19 (object file identifier), or Key 23 (object journal identifier) may be specified, but not both.
4. Hexadecimal zero is not valid.
5. The Get Attributes Qp0lGetAttr() API may be used to retrieve the journal identifier for an object.

«

Object path. A list of integrated file system objects, entered via path name, for which journal entries are to be retrieved. Only objects of type *STMF, *DIR, or *SYMLNK that are in the ″root″ (/), QOpenSys, and user-defined file systems are supported. All other objects are ignored. For the format of this field, see “Object Path Format” on page 79.

Only objects that are currently linked with the specified path name and have a journal identifier associated with them are used in journal entry selection. If the specified object does exist, the journal identifier associated with that link is used for journal entry selection. If a specified object does not exist or does not have a journal identifier associated with it, that link is not used in selecting journal entries and no error is sent.

Notes:

1. The sum of the lengths of all the variable length records that precede this key must be a multiple of 16 bytes. If not, errors will occur. For ease of implementation, it is recommended that the length of each variable length record be a mulitple of 16 bytes.
2. The journal identifier is the unique identifier associated with the object when journaling is started for that object. The journal identifier stays constant, even if the object is renamed, moved or restored. See the Journal management topic collection for more information.
3. When specifying an object on this key, journal entries with the following journal code values are retrieved only if they satisfy the values specified on the other keys:
   - Journal code B (integrated file system information entries).
   - Journal code U (user-generated entries).
   - Other journal codes if *IGNOBJSLT is the second element of the journal code key. If *ALLSLT is the second element of the journal code key, no journal entries with that code are retrieved.
4. Either Key 16 (file) may be specified, or one or more of the object keys, Key 17 (object), Key 18 (object path), Key 19 (object file identifier), » or Key 23 (object journal identifier)« may be specified, but not both.

Program. Whether the journal entries being retrieved are limited to the journal entries for a specified program. Only journal entries for the specified program name are considered for retrieval. The default is *ALL. The possible values are:

**ALL*	The retrieval is not limited to entries for a specific program.
program	The name of the program whose journal entries are considered for retrieval. Only journal entries for this program are considered for retrieval.

Range of journal receivers. The qualified names of the starting (first) and ending (last) journal receivers used in the search for a journal entry to be retrieved. For the format of this field, see “Receiver Range Format” on page 80. The system starts the search with the starting journal receiver and proceeds through the receiver chain until the ending journal receiver is processed.

If *CURRENT, *CURCHAIN, » or *CURAVLCHN « is specified for the starting journal receiver, the remaining fields should be set to blanks. The default is *CURRENT for the starting journal receiver. If the total number of receivers in the range is larger than 2045, an error message is sent and no journal entry is retrieved.

Starting sequence number. The first journal entry considered for retrieval. This Char(20) field is treated as Zoned(20,0) except when the special value *FIRST is specified. The default is *FIRST. The possible values are:

**FIRST*	The first journal entry in the specified journal receiver range is the first entry considered for retrieval.

sequence number The sequence number of the journal entry that is the first entry considered for retrieval.

Note: If this key is specified, Key 3 (starting time stamp) cannot also be specified.

Starting time stamp. The time stamp of the first journal entry considered for retrieval. This Char(26) field is in the format YYYY-MM-DD-HH.MM.SS.UUUUUU where

YYYY	Year
MM	Month
DD	Day
HH	Hours
MM	Minutes
SS	Seconds
UUUUUU	Microseconds

Notes:

1. If this key is specified, Key 2 (starting sequence number) cannot also be specified.
2. Note: If the system value QLEAPADJ (Leap year adjustment) is zero, then the result returned will be in 1 microsecond granularity. If the system value QLEAPADJ is greater than zero, then the result returned will be in 8 microsecond granularity.

User profile. Whether the journal entries being retrieved are limited to the journal entries for a specified user profile name. The user profile name is the user profile under which the job is run that deposited the journal entries. Only journal entries for the specified user profile are considered for retrieval. The default is *ALL. The possible values are:

**ALL*	The retrieval is not limited to entries for a specific user profile.
user profile	The name of the user profile whose journal entries are considered for retrieval. Only journal entries for this user profile are considered for retrieval.

File Format

Offset			
Dec	**Hex**	**Type**	**Field**
		BINARY(4)	Number in array
Note: These fields repeat for each file member.			
		CHAR(10)	File name
		CHAR(10)	Library name
		CHAR(10)	Member name

Field Descriptions

File name. The file name for which journal entries are to be retrieved. The possible values are:

**ALLFILE*	The search for the journal entries retrieved is not limited to a specified file name. All journal entries are converted for output, regardless of which objects, if any, the entries are associated with. If *ALLFILE is specified, the associated library name and member name fields should be blank.
**ALL*	Journal entries for all physical files in the specified library (the library name must be specified) for which journaled changes currently in the specified journal receiver range are retrieved. If *ALL is specified and the user does not have the required authority to all of the files, an error occurs and the command ends.

file name The name of the physical database file for which journaled changes are being retrieved.

Library name. The library name associated with the file name for which journal entries are to be retrieved. 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.
blank The library name field must be blank if *ALLFILE is specified for the file name.

Member name. The file member name for which journal entries are to be retrieved. The possible values are:

**FIRST* Entries for the database file and the first member in the file are retrieved. This value is not valid when the journal is a remote journal.
**ALL* Entries for the database file and all the currently existing members of the file are retrieved.
**NONE* Only entries for the database file are retrieved. Entries for members of the file are not retrieved.
member name The name of the file member for which entries are retrieved. If the specified physical file does not exist on the system, specify either *ALL or a specific file member name.

If *ALL is specified for the file name, this member name is used for all applicable files in the library.
blank The member name field must be blank if *ALLFILE is specified for the file name.

Number in array. The number of file codes that are specified for this key. The possible values are 1 through 300. The value must be 1 if *ALLFILE or *ALL is specified for file name.

Journal Code Format

Offset		Type	Field
Dec	**Hex**		
		BINARY(4)	Number in array
Note: These fields repeat for each journal code.			
		CHAR(10)	Journal code value
		CHAR(10)	Journal code selection

Field Descriptions

Journal code value. The journal code for which journal entries are to be retrieved. The possible values are:

**ALL* The retrieval of journal entries is not limited to entries with a particular journal code.
**CTL* Only journal entries deposited to control the journal functions are to be retrieved (journal codes = J and F).
code The 1-character journal code for which journal entries are to be retrieved.

A list of journal codes that can be specified is provided in the Journal management topic collection. The 1-character code should be left-justified.

Journal code selection. Whether other selection criteria apply to this specified journal code. The possible values are:

**ALLSLT* The journal entries with the specified journal code are to be retrieved only if all selection keys are satisfied.

IGNFILSLT* The journal entries with the specified journal code are to be retrieved only if all selection keys except the file key are satisfied. **Note: This value is not valid for journal codes D, F, and R. This value is not valid if Key 17 (object), Key 18 (object path), Key 19 (object file identifier), or Key 23 (object journal identifier) is specified.

IGNOBJSLT* The journal entries with the specified journal code are to be retrieved only if all selection keys are satisfied except the object, object path, object file identifier, directory subtree, name pattern, and object journal identifier keys. **Note: This value is not valid for journal codes B, D, E, F, Q, R, and Y. This value is not valid if Key 16 (file) is specified.

blank The journal code selection must be blank if *ALL or *CTL is specified for the journal code value.

Number in array. The number of journal codes that are specified for this key. The possible values are 1 through 16. The value must be 1 if *ALL or *CTL is specified for the journal code value.

Journal Entry Type Format

Offset		Type	Field
Dec	**Hex**		
		BINARY(4)	Number in array
Note: These fields repeat for each entry type.			
		CHAR(10)	Journal entry type

Field Descriptions

Journal entry types. The journal entry types for which journal entries are to be retrieved. The possible values are:

**ALL* The retrieval of journal entries is not limited to entries with a particular journal entry type.

**RCD* Only journal entries that have an entry type for record level operations are retrieved. The following entry types are valid: BR, DL, DR, IL, PT, PX, UB, UP, and UR.

entry type The 2-character entry type that limits the search for the journal entries to retrieve. Only journal entries that contain the specified entry type are considered for retrieval. A list of valid entry types is in the Journal management topic collection. The 2-character entry type should be left-justified.

Number in array. The number of journal entry types that are specified for this key. The possible values are 1 through 300. The value must be 1 if *ALL or *RCD is specified for journal entry type.

Name Pattern Format

Offset		Type	Field
Dec	**Hex**		
		BINARY(4)	Number in array
4	4	CHAR(12)	Reserved
Note: These fields repeat for each pattern.			
16	10	BINARY(4)	Length of this pattern entry

Offset		Type	Field
Dec	Hex		
		CHAR(10)	Include or omit
		CHAR(2)	Reserved
		PTR(16)	Pointer to pattern path structure

Field Descriptions

Include or omit. Whether the name pattern is included or omitted from the retrieve operation.

**INCLUDE* The objects that match the object name pattern are to be included in determining what journal entries are retrieved, unless overridden by an *OMIT specification.

**OMIT* The objects that match the object name pattern are not to be included in determining what journal entries are retrieved. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Length of this pattern entry. The length of the current pattern entry that can be used as the displacement from the start of this pattern entry to the next pattern entry. The length must be a minimum of 32 bytes and must be a multiple of 16.

Number in array. The number of patterns that are specified for this key. The possible values are 1 through 20.

Pointer to pattern path structure. A pointer to a path structure.

This pointer must be 16-byte aligned. If not, unpredictable results may occur.

Additional information about path name patterns is in the Integrated file system topic collection.

The pointer given points to a path name structure. If that path name structure contains a pointer, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the pattern path name format, see Path name format.

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

Object Format

Offset		Type	Field
Dec	Hex		
		BINARY(4)	Number in array
Note: These fields repeat for each object.			
		CHAR(10)	Object name
		CHAR(10)	Library name
		CHAR(10)	Object type
		CHAR(10)	Member name, if *FILE specified

Field Descriptions

Library name. The library name associated with the object for which journal entries are to be retrieved. 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.

Member name, if *FILE specified. The file member name for which journal entries are to be retrieved. If the specified object type was not *FILE, the member name is ignored. The possible values are:

**FIRST*	Entries for the database file and the first member in the file are retrieved. This value is not valid when the journal is a remote journal.
**ALL*	Entries for the database file and all the currently existing members of the file are retrieved.
**NONE*	Only entries for the database file are retrieved. Entries for members of the file are not retrieved.
member name	The name of the file member for which entries are retrieved. If the specified physical file does not exist on the system, specify either *ALL or a specific file member name. If *ALL is specified for the file name, this member name is used for all applicable files in the library.

Number in array. The number of object names that are specified for this key. The possible values are 1 through 300.

Object name. The object name for which journal entries are to be retrieved. The possible values are:

**ALL*	Journal entries for all objects of the specified object type in the specified library (the library name must be specified) for which journaled changes currently in the specified journal receiver range are retrieved. If *ALL is specified and the user does not have the required authority to all of the objects, an error occurs and the command ends.
object name	The name of the object for which journaled changes are being retrieved.

Object type. The object type associated with the object for which journal entries are to be retrieved. The possible values are:

**FILE*	Entries for database files and database file members are retrieved.
**DTAARA*	Entries for data areas are retrieved.
**DTAQ*	Entries for data queues are retrieved.
**LIB*	Entries for libraries are retrieved.

Object File Identifier Format

Offset		Type	Field
Dec	**Hex**		
		BINARY(4)	Number in array
Note: These fields repeat for each object.			
		CHAR(16)	File identifier

Field Descriptions

File identifier. The file identifier of the object for which journal entries are to be retrieved.

Number in array. The number of file identifiers that are specified for this key. The possible values are 1 through 300.

Object Journal Identifier Format

Offset			
Dec	**Hex**	**Type**	**Field**
		BINARY(4)	Number in array
Note: These fields repeat for each object.			
		CHAR(10)	Journal identifier

Field Descriptions

Journal identifier. The journal identifier of the object for which journal entries are to be retrieved.

Number in array. The number of journal identifiers that are specified for this key. The possible values are 1 through 300.

Object Path Format

Offset			
Dec	**Hex**	**Type**	**Field**
		BINARY(4)	Number in array
		CHAR(12)	Reserved
Note: These fields repeat for each object path name.			
		BINARY(4)	Length of this object path name entry
		CHAR(10)	Include or omit
		CHAR(2)	Reserved
		PTR(16)	Pointer to an object path name

Field Descriptions

Include or omit. Whether names that match the path name should be included or omitted from the operation. Note that in determining whether a name matches a pattern, relative name patterns are always treated as relative to the current working directory.

Note: Key 20 (directory subtree) specifies whether the subtrees are included or omitted.

**INCLUDE* The objects that match the object name pattern are to be included in determining what journal entries are retrieved, unless overridden by an *OMIT specification.

**OMIT* The objects that match the object name pattern are not to be included in determining what journal entries are retrieved. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Length of this object path name entry. The length of the current object path name entry that can be used as the displacement from the start of this path name entry to the next path name entry. The length must be a minimum of 32 bytes and must be a multiple of 16.

Number in array. The number of object path names that are specified for this key. The possible values are 1 through 300.

Pointer to an object path name. A pointer to the object path name of the object for which journal entries are to be retrieved. All path names are relative to the current directory at the time of the call.

In the last component of the path name, an asterisk (*) or a question mark (?) can be used to search for patterns of names. The * tells the system to search for names that have any number of characters in the position of the * character. The ? tells the system to search for names that have a single character in the position of the ? character. Symbolic links within the path name will not be followed. If the path name begins with the tilde (~) character, then the path is assumed to be relative to the appropriate home directory.

Additional information about path name patterns is in the Integrated file system topic collection.

The pointer given points to a path name structure. If that path name structure contains a pointer, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the path name format, see Path name format.

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

Receiver Range Format

Offset		Type	Field
Dec	Hex		
		CHAR(10)	Starting journal receiver name
		CHAR(10)	Starting journal receiver library
		CHAR(10)	Ending journal receiver name
		CHAR(10)	Ending journal receiver library

Field Descriptions

Ending journal receiver library. The ending journal receiver library for which journal entries are to be retrieved. 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	The name of the library to be searched.
blank	This field can be blank only if *CURRENT, *CURCHAIN, » or *CURAVLCHN « is specified.

Ending journal receiver name. The ending journal receiver name for which journal entries are to be retrieved. The possible values are:

**CURRENT*	The journal receiver that is attached when starting to retrieve journal entries is used. If *CURRENT is specified, the associated library name field should be blank.

name The name of the last journal receiver that contains entries to be retrieved. If a name is specified, the ending journal receiver name field must be specified also. If the end of the receiver chain is reached before a receiver of this name is found, an error message is sent and no journal entry is retrieved.

blank This field can be blank only if *CURRENT, *CURCHAIN, » or *CURAVLCHN « is specified.

Starting journal receiver library. The starting journal receiver library for which journal entries are to be retrieved. 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 The name of the library to be searched.

blank This field can be blank only if *CURRENT, *CURCHAIN, » or *CURAVLCHN « is specified.

Starting journal receiver name. The starting journal receiver name for which journal entries are to be retrieved. The possible values are:

**CURRENT* The journal receiver that is attached when starting to retrieve journal entries is used. If *CURRENT is specified, the associated library name and ending journal receiver fields should be blank.

**CURCHAIN* The journal receiver chain that includes the journal receiver that is attached when starting to retrieve journal entries is used. This receiver chain does not cross a break in the chain. If there is a break in the chain, the receiver range is from the most recent break in the chain through the receiver that is attached when starting to retrieve journal entries. If *CURCHAIN is specified, the associated library name and ending journal receiver fields should be blank.

Note: For journal receivers with reset sequence numbers in the chain, the QjoRetrieveJournalEntries API may return the same journal entries for repeated API calls. To avoid receiving the same journal entries, change the starting journal receiver name field to indicate the next receiver in the chain after the initial call to the API. The reset sequence numbers do not cause a break in the journal receiver chain, but rather the sequence number is reset to one at the beginning of a new journal receiver. See the restrictions that discuss the reset sequence number.

» **CURAVLCHN* The journal receiver chain that includes the journal receiver that is attached when starting to retrieve journal entries is used. This receiver chain does not cross a break in the chain. If there is a break in the chain, the receiver range is from the most recent break in the chain through the receiver that is attached when starting to retrieve journal entries. If journal receivers exist in the receiver chain that are not available because they were saved with the storage freed option, those journal receivers will be ignored and entries will be retrieved starting with the first available journal receiver in the chain. If *CURAVLCHN is specified, the associated library name and ending journal receiver fields should be blank. «

name The name of the first journal receiver that contains entries to be retrieved.

RJNE0100 Format

The structure of the information returned is determined by the specified format name. For detailed descriptions of the fields, see "Field Descriptions" on page 86. The retrieved data is composed of four different sections as follows:

- A header section. Only one header section is returned per call. See Header (page 82).
- Journal entry sections. These three sections will be repeated for each journal entry retrieved.
  - Header section of journal entry. See This journal entry's header with format RJNE0100 (page 82).
  - Null value indicators section of journal entry. This section will be one of the two following formats, depending on what was specified for the null value indicators key.

- If the user did not specify the null value indicators key or specified null value indicators length(*VARLEN), see This journal entry's null value indicators if Null Value Indicators (*VARLEN) specified (page 83).
- If the user specified null value indicators length(field length), see This journal entry's null value indicators if Null Value Indicators (field length) specified (page 83).

  Note: If a null value indicators length of 0 was specified, then this section will not appear in the journal entry data.

– Entry specific data section of journal entry. See This journal entry's entry specific data (page 83).

Header

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Offset to first journal entry header
8	8	BINARY(4)	Number of entries retrieved
12	C	CHAR(1)	Continuation handle

This journal entry's header with format RJNE0100

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Displacement to next journal entry's header
4	4	BINARY(4)	Displacement to this journal entry's null value indicators
8	8	BINARY(4)	Displacement to this journal entry's entry specific data
12	C	BINARY(4), UNSIGNED	Pointer handle
16	10	CHAR(20)	Sequence number
36	24	CHAR(1)	Journal code
37	25	CHAR(2)	Entry type
39	27	CHAR(26)	Time stamp
65	41	CHAR(10)	Job name
75	4B	CHAR(10)	User name
85	55	CHAR(6)	Job number
91	5B	CHAR(10)	Program name
101	65	CHAR(30)	Object
131	83	CHAR(10)	Count/relative record number
141	8D	CHAR(1)	Indicator flag
142	8E	CHAR(20)	Commit cycle identifier
162	A2	CHAR(10)	User profile
172	AC	CHAR(8)	System name
180	B4	CHAR(10)	Journal identifier
190	BE	CHAR(1)	Referential constraint
191	BF	CHAR(1)	Trigger
192	C0	CHAR(1)	Incomplete data

Offset		Type	Field
Dec	Hex		
193	C1	CHAR(1)	Object name indicator
194	C2	CHAR(1)	Ignore during APYJRNCHG or RMVJRNCHG
195	C3	CHAR(1)	Minimized entry specific data

This journal entry's null value indicators if Null Value Indicators (*VARLEN) specified

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of null value indicators
4	4	CHAR(*)	Null value indicators

This journal entry's null value indicators if Null Value Indicators (field length) specified

Offset		Type	Field
Dec	Hex		
0	0	CHAR(specified Null value indicators field length)	Null value indicators

This journal entry's entry specific data

Offset		Type	Field
Dec	Hex		
0	0	CHAR(5)	Length of entry specific data
5	5	CHAR(11)	Reserved
16	16	CHAR(*)	Entry specific data

RJNE0200 Format

The structure of the information returned is determined by the specified format name. For detailed descriptions of the fields, see "Field Descriptions" on page 86. The retrieved data is composed as follows:

- A header section. Only one header section is returned per call. See Header (page 84).
- Journal entry sections. These optional sections will be repeated for each journal entry retrieved.
  - Header section of journal entry. See This journal entry's header with format RJNE0200 (page 84).
  - Transaction identifier section of journal entry if the displacement to transaction identifier is not 0. See the QSYSINC/H.XA header file for the layout of this data.
  - Logical unit of work section of journal entry if the displacement to logical unit of work is not 0. See This journal entry's Logical unit of work if the displacement to logical unit of work is not 0 (page 86).
  - Receiver information section of journal entry if the displacement to receiver information is not 0. See This journal entry's receiver information if the displacement to receiver information is not 0 (page 86).

– Null value indicators section of journal entry if the displacement to null values indicators is not 0. This section will be one of the two following formats, depending on what was specified for the null value indicators key
  - If the user did not specify the null value indicators key or specified null value indicators length(*VARLEN), see This journal entry's null value indicators if Null Value Indicators (*VARLEN) specified (page 83).
  - If the user specified null value indicators length(field length), see This journal entry's null value indicators if Null Value Indicators (field length) specified (page 83).

    Note: If a null value indicators length of 0 was specified, then this section will not appear in the journal entry data.
– Entry specific data section of journal entry if the displacement to entry specific data is not 0. See This journal entry's entry specific data (page 83).

Header

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Offset to first journal entry header
8	8	BINARY(4)	Number of entries retrieved
12	C	CHAR(1)	Continuation indicator
13	D	CHAR(10)	Continuation starting receiver
23	17	CHAR(10)	Continuation starting receiver library
33	21	CHAR(20)	Continuation starting sequence number
53	35	CHAR(11)	Reserved

This journal entry's header with format RJNE0200

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4), UNSIGNED	Displacement to next journal entry's header
4	4	BINARY(4), UNSIGNED	Displacement to this journal entry's null value indicators
8	8	BINARY(4), UNSIGNED	Displacement to this journal entry's entry specific data
12	C	BINARY(4), UNSIGNED	Displacement to this journal entry's transaction identifier
16	10	BINARY(4), UNSIGNED	Displacement to this journal entry's logical unit of work
20	14	BINARY(4), UNSIGNED	Displacement to this journal entry's receiver information
24	18	BINARY(8), UNSIGNED	Sequence number
32	20	BINARY(8), UNSIGNED	Unformatted Time stamp
40	28	BINARY(8), UNSIGNED	Thread identifier

Offset		Type	Field
Dec	Hex		
48	30	BINARY(8), UNSIGNED	System sequence number
56	38	BINARY(8), UNSIGNED	Count/relative record number
64	40	BINARY(8), UNSIGNED	Commit cycle indentifier
72	48	BINARY(4), UNSIGNED	Pointer handle
76	4C	BINARY(2), UNSIGNED	Remote port
78	4E	BINARY(2), UNSIGNED	Arm number
80	50	BINARY(2), UNSIGNED	Program library ASP number
82	52	CHAR(16)	Remote Address
98	62	CHAR(1)	Journal code
99	63	CHAR(2)	Entry type
101	65	CHAR(10)	Job name
111	6F	CHAR(10)	User name
121	79	CHAR(6)	Job number
127	7F	CHAR(10)	Program name
137	89	CHAR(10)	Program library name
147	93	CHAR(10)	Program library ASP device name
157	9D	CHAR(30)	Object
187	BB	CHAR(10)	User profile
197	C5	CHAR(10)	Journal identifier
207	CF	CHAR(1)	Address family
208	D0	CHAR(8)	System name
216	D8	CHAR(1)	Indicator flag
217	D9	CHAR(1)	Object name indicator
218(0)	DA(0)	BIT(1)	Referential constraint
218(1)	DA(1)	BIT(1)	Trigger
218(2)	DA(2)	BIT(1)	Incomplete data
218(3)	DA(3)	BIT(1)	Ignored during APYJRNCHG or RMVJRNCHG
218(4)	DA(4)	BIT(1)	Minimized entry specific data
218(5)	DA(5)	BIT(1)	File type indicator
218(6)	DA(6)	BIT(1)	Minimized on field boundaries
218(7)	DA(7)	BIT(1)	Reserved
219	DB	CHAR(10)	Object type
229	E5	CHAR(3)	Reserved
229	E8	BINARY(4), UNSIGNED	Nested commit level

This journal entry's Logical unit of work if the displacement to logical unit of work is not 0

Offset		Type	Field
Dec	Hex		
0	0	CHAR(39)	Logical unit of work

This journal entry's receiver information if the displacement to receiver information is not 0

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Receiver name
10	A	CHAR(10)	Receiver library name
20	14	CHAR(10)	Receiver library ASP device name
30	1E	BINARY(2)	Receiver library ASP number

Field Descriptions

Address family. The address family identifies the format of the remote address for this journal entry. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*RMTADR) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then 0 will be returned for the address family.

0 This entry was not associated with any remote address.
4 The format of the remote address is internet protocol version 4. The remote address is returned as a 16-byte character field.
6 The format of the remote address is internet protocol version 6. The remote address is returned as a 128-bit binary number.

Arm number. The number of the disk arm that contains the journal entry.

Bytes returned. The number of bytes of data returned.

If an error message is returned, other than error messages CPF3CF1, CPF3C90, CPF6948, CPF6949 or CPF9872, this field should be checked to determine if partial journal entry information has been returned.

Commit cycle identifier. A number that identifies the commit cycle. This is either a Char(20) or Binary(8) field and if Char(20), it is treated as Zoned(20,0). A commit cycle is from one commit or rollback operation to another.

The commit cycle identifier is found in every journal entry that is associated with a commitment transaction. If the journal entry was not made as part of a commitment transaction, this field is zero.

Continuation handle. An indicator for more journal entries available that meet any specified selection criteria. The possible values are:

0 All the journal entries that match the search criteria are returned to this structure.
1 There are more journal entries available in the specified receiver range that match the search criteria, but there is no room available in the return structure. You may request more data by calling the API again, and by specifying one more than the sequence number of the last journal entry returned as the starting sequence number on the next API call as long as there has been no reset of the sequence number within the receiver range.

Note: If an error message was returned and partial journal entry information was returned, this field may not correctly indicate whether additional journal entries are available.

Continuation indicator. An indicator for more journal entries available that meet the specified selection criteria. The possible values are:

0 All the journal entries that match the search criteria are returned to this structure.

1 There are more journal entries available in the specified receiver range that match the search criteria, but there is no room available in the return structure. You may request more data by calling the API again, and by specifying the following as part of your selection criteria:

Starting receiver name
Set from the value returned in Continuation starting receiver.

Starting receiver library name
Set from the value returned in Continuation starting receiver library.

Starting sequence number
Set from the value returned in Continuation starting sequence number.

Note: If an error message was returned and partial journal entry information was returned, this field may not correctly indicate whether additional journal entries are available.

Continuation starting receiver library. When the continuation indicator is *1*, then this field will identify the name of the library that contains the receiver that holds the next journal entry that could be retrieved with the same selection criteria on a subsequent call to this API. When used in conjunction with the continuation starting receiver name and the continuation starting sequence number, a subsequent API call will ensure that no journal entries in the given receiver range will be skipped, irrespective of any reset of sequence numbers that may have taken place within the given receiver range. When the continuation indicator is *0*, then this field will be blanks.

Continuation starting receiver. When the continuation indicator is *1*, then this field will identify the name of the receiver that holds the next journal entry that could be retrieved with the same selection criteria on a subsequent call to this API. When used in conjunction with the continuation starting receiver library name and the continuation starting sequence number, a subsequent API call will ensure that no journal entries in the given receiver range will be skipped, irrespective of any reset of sequence numbers that may have taken place within that receiver range. When the continuation indicator is *0*, then this field will be blanks.

Continuation starting sequence number. When the continuation indicator is *1*, then this field will identify the sequence number of the next journal entry that could be retrieved with the same selection criteria on a subsequent call to this API. When used in conjunction with the continuation starting receiver library name and the continuation starting receiver name, a subsequent API call will ensure that no journal entries in the given receiver range will be skipped, irrespective of any reset of sequence numbers that may have taken place within that receiver range. When the continuation indicator is *0*, then this field will be blanks. This is a Char(20) field that is treated as Zoned(20,0).

Count/relative record number. Contains either the relative record number (RRN) of the record that caused the journal entry or a count that is pertinent to the specific type of journal entry. See the Journal Entry Information appendix in the Journal management topic collection to see specific values for this field, if applicable. This is either a Char(10) or a unsigned Binary(8) field and if Char(10), it is treated as Zoned(10,0).

Displacement to next journal entry's header. The displacement from the start of this journal entry's header section to the start of the journal entry header section for the next journal entry.

Displacement to this journal entry's entry specific data. The displacement from the start of this journal entry's header section to the start of the entry specific data section for this journal entry. A value of 0 indicates that this data is not returned for this journal entry.

Displacement to this journal entry's logical unit of work. The displacement from the start of this journal entry's header section to the start of the logical unit of work section for this journal entry. A value of 0 indicates that this data is not returned for this journal entry.

Displacement to this journal entry's receiver information. The displacement from the start of this journal entry's header section to the start of the receiver information section for this journal entry. A value of 0 indicates that this data is not returned for this journal entry. Journal receiver information is returned only for the first entry in a buffer and when the receiver information changes from one journal entry to the next. If no journal receiver information is returned, it can be assumed that the receiver information from the previous entry will apply to the current journal entry.

Displacement to this journal entry's null value indicators. The displacement from the start of this journal entry's header section to the start of the null value indicators section for this journal entry. A value of 0 indicates that this data is not returned for this journal entry.

Displacement to this journal entry's transaction identifier. The displacement from the start of this journal entry's header section to the start of the transaction identifier section for this journal entry. A value of 0 indicates that this data is not returned for this journal entry.

Entry specific data. The entry specific data returned for this journal entry. See the Journal management topic collection for the layouts of this information for each journal entry type.

If the incomplete data indicator is on, then this data contains pointers to additional journal entry data. See "Use of Pointers within Entry Specific Data" on page 94 for a discussion on the use of these pointers.

Entry type. Further identifies the type of user-created or system-created entry. See the journal entry information in the Journal management topic collection for descriptions of the entry types.

Entry type. Further identifies the type of user-created or system-created entry. See the journal entry information in the Journal management topic collection for descriptions of the entry types.

File type indicator. Identifies whether or not this journal entry is associated with a logical file. The value will be 0 if the value for object type is not *FILE. The possible values are:

0 This entry is not associated with a logical file.
1 This entry is associated with a logical file.

Ignore during APYJRNCHG or RMVJRNCHG. Whether this entry is ignored during a Apply Journaled Changes (APYJRNCHG) or Remove Journaled Changed (RMVJRNCHG) command. The possible values are:

0 This entry will not be ignored during APYJRNCHG or RMVJRNCHG
1 This entry will be ignored during APYJRNCHG or RMVJRNCHG

Incomplete data. Whether this entry has data that must be additionally retrieved using a pointer returned for the missing information. See "Use of Pointers within Entry Specific Data" on page 94 for more information. The possible values are:

0 This entry does not have any pointers included.
1 This entry does have pointers included.

Indicator flag. An indicator for the operation. See the journal entry information in the Journal management topic collection to see specific values for this field, if applicable.

Job name. The name of the job that added the entry.

Notes:

1. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*JOB) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then *OMITTED is returned for the job name.
2. If the journal entry was deposited by a system task that was not associated with a job, then *TDE will be returned for the job name.
3. If the job name was not available when the journal entry was deposited, then *NONE is returned for the job name.

Job number. The job number of the job that added the entry.

Notes:

1. If the RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*JOB) was not was in effect for the journal when the journal receiver that contains the journal entry was attached, then zeros are returned for the job number.
2. If the journal entry was deposited by a system task that was not associated with a job, then zeros will be returned for the job number.
3. If the job name was not available when the journal entry was deposited, then zeros are returned for the job number.

Journal code. The primary category of the journal entry. See the Journal Entry Information section in the Journal management topic collection for descriptions of the journal codes.

Journal identifier. The journal identifier (JID) for the object. When journaling is started for an object, the system assigns a unique JID to that object. The JID remains constant even if the object is renamed or moved. If journaling is stopped, however, there is no guarantee that the JID will be the same when journaling is started again for the same object.

If no JID is associated with the entry, this field has hexadecimal zeros.

Length of entry specific data. The length of the entry specific data returned for this journal entry. This Char(5) field is treated as Zoned(5,0). If the entry specific data includes any pointers to additional data, the length of that additional data in not included in this value. See "Use of Pointers within Entry Specific Data" on page 94 for more information.

Length of null value indicators. The length of the null value indicators returned for this journal entry.

Logical unit of work. The logical unit of work identifies entries to be associated with a given unit of work, usually within a commit cycle. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*LUW) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then no logical unit of work will be returned for this entry and the displacement to this entry's logical unit of work will be 0.

Minimized entry specific data. Whether this entry has minimized entry specific data as a result of the journal having specified MINENTDTA for the object type of the entry. The possible values are:

0 This entry has complete entry specific data.
1 This entry has minimized entry specific data.

Minimized on field boundaries. Whether this entry has minimized entry specific data on field boundaries as a result of the journal having been specified with MINENTDTA(*FLDBDY). The possible values are:

0 This entry does not have minimized entry specific data on field boundaries.
1 This entry has minimized entry specific data on field boundaries. Therefore, the entry specific data can be viewable and may be used for auditing purposes. In order for the entry specific data to be viewable, a value of *YES must have be specified on Key 22 (format minimized data). If a value of *YES was specified on Key 22, then the fields that were changed are accurately reflected. The fields that were not changed and were not recorded return default data and are indicated by a value of '9' in the null value indicators field.

Nested commit level. Indicates the nesting level of the commit cycle that was open when a journal entry representing an object level change was deposited. The primary commit cycle is considered the first level of nesting and subsequent save point entries that were deposited prior to this entry correspond to additional levels of nesting. This field will be zero if any of the following are true:

- This journal entry does not represent an object level change (object level changes are the result of using commands like: CRTPF, CHGPF, MOVOBJ, and RNMOBJ).
- This journal entry was not deposited under commitment control.
- This journal entry was deposited on a release prior to V5R4M0.

Null value indicators. The null value indicators returned for this journal entry. If the record image has not been minimized or has been minimized on field boundaries in the entry specific data, then there is one null value indicator for each field in the physical file. Each indicator is one character long and has one of the following values:

0 Corresponding field is not null.
1 Corresponding field is null.
9 Corresponding field was not collected and is represented with default data.

Number of entries retrieved. The number of journal entries that were retrieved.

If an error message is returned, other than error messages CPF3CF1, CPF3C90, CPF6948, CPF6949 or CPF9872, a non-zero bytes returned field will reflect how much data was returned prior to the sending of the error message.

Object. The name of the object for which the journal entry was added. If the entry is not associated with a journaled object, this field is blank.

If the object associated with the journal entry is a file object the format of this field is:

Char(10)	File name
Char(10)	File library name
Char(10)	Member name

Note: If the journal receiver was attached prior to installing V4R2M0 on your system, the following items are true:

- If *ALLFILE is specified for the file key, then the fully qualified name is the most recent name of the file when the newest receiver in the receiver range was the attached receiver and when the file was still being journaled.
- If a file name is specified or if library *ALL is specified on the file key, the current fully qualified name of the file appears in the retrieved journal entry.

If the journal receiver was attached while V4R2M0 or a later release was running on the system, the fully qualified name is the name of the object at the time the journal entry was deposited.

If the object associated with the journal entry is an integrated file system object, the format of this field is:

Char(16)	File identifier
Char(14)	Blanks

For all other entries associated with journaled objects, the format of this information is:

Char(10)	Object name
Char(10)	Object library name
Char(10)	Blanks

Object name indicator. An indicator with respect to the information in the object field. The valid values are:

0 Either the journal entry has no object information or the object information in the journal entry header does not necessarily reflect the name of the object at the time the journal entry was deposited into the journal.

Note: This value is returned only when retrieving journal entries from a journal receiver that was attached to a journal prior to V4R2M0.

1 The object information in the journal entry header reflects the name of the object at the time the journal entry was deposited into the journal.

2 The object information in the journal entry header does not necessarily reflect the name of the object at the time the journal entry was deposited into the journal. The object information may be returned as a previously known name for the object prior to the journal entry being deposited into the journal or be returned as *UNKNOWN.

Note: This value will be returned only when retrieving journal entries from a remote journal and the remote journal is currently being caught up from its source journal. A remote journal is being caught up from its source journal when the Change Remote Journal (CHGRMTJRN) command or Change Journal State (QjoChangeJournalState) API is called and is currently replicating journal entries to the remote journal. After the call to the CHGRMTJRN command or QjoChangeJournalState API returns, the remote journal is maintained with a synchronous or asynchronous delivery mode, and the remote journal is no longer being caught up.

» 3 The object information in the journal entry header does not necessarily reflect the name of the object at the time the journal entry was deposited into the journal. The object information may be returned as *UNKNOWN. «

Object type. The type of object in the entry. The possible values are:

**DIR*	This entry is for an integrated file system directory.
**DTAARA*	This entry is for a data area.
**DTAQ*	This entry is for a data queue.
**FILE*	This entry is for a database file.
**JRNRCV*	This entry is for a journal receiver.
» **LIB*	This entry is for a library. «
**QDDS*	This entry is for the data portion of a database member.
**QDDSI*	This entry is for an access path of a database member.
**STMF*	This entry is for an integrated file system stream file.
**SYMLNK*	This entry is for an integrated file system symbolic link.

Offset to first journal entry header. The offset from the start of the format to the journal entry header section for the first journal entry that is retrieved. If no entries are retrieved, this value is 0.

Pointer handle. If the entry specific data returned for this journal entry returned any pointers, this is the handle associated with those pointers. Otherwise, it is 0.

See "Use of Pointers within Entry Specific Data" on page 94 for a discussion on the use of these pointers and what you must do with this pointer handle.

Program library ASP device name. The name of the ASP device that contains the program.

Notes:

1. If the program library ASP is not an independent ASP, then *SYSBAS will be returned for the program library ASP device name.
2. If the program library ASP device name was not available when the journal entry was deposited, or if RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*PGMLIB) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then *OMITTED is returned for the program library ASP device name.

Program library ASP number. The number for the auxiliary storage pool that contains the program that added the journal entry. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*PGMLIB) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then Hex 0 will be returned for program ASP number.

Program library name. The name of the library that contains the program that added the journal entry. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*PGMLIB) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then *OMITTED will be returned for the program library name.

Program name. The name of the program that added the entry. If an application or CL program did not add the entry, the field contains the name of a system-supplied program such as QCMD or QPGMMENU. If the program name is the special value *NONE, then one of the following is true:

- The program name does not apply to this journal entry.
- The program name was not available when the journal entry was made. For example, the program name is not available if the program was destroyed.

If the program that deposited the journal entry is an original program model program, this data will be complete. Otherwise, this data is unpredictable.

If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*PGM) was not in effect for the journal when the journal receiver that contains this journal entry was attached, *OMITTED is returned as the program name.

Receiver library ASP device name. The name of the ASP device that contains the receiver.

Notes:

1. If the receiver library ASP is not an independent ASP, then *SYSBAS will be returned for the receiver library ASP device name.
2. If the receiver library ASP device name was not available when the journal entry was deposited, then *OMITTED is returned for the receiver library ASP device name.

Receiver library ASP number. The number for the auxiliary storage pool containing the receiver holding the journal entry.

Receiver library name. The name of the library containing the receiver holding the journal entry.

Receiver name. The name of the receiver holding the journal entry.

Referential constraint. Whether this entry was recorded for actions that occurred on records that are part of a referential constraint.

0 This entry was not created as part of a referential constraint.
1 This entry was created as part of a referential constraint.

Remote address. The remote address associated with the journal entry. The format of the address is dependent on the value of the address family for this journal entry. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*RMTADR) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then Hex 0 will be returned for remote address.

Remote port. The port number of the remote address associate with this journal entry. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*RMTADR) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then Hex 0 will be returned for remote port.

Reserved. Reserved area. It always contains hexadecimal zeros.

Sequence number. A number assigned by the system to each journal entry. This is either a Char(20) or Binary(8) field and if Char(20), it is treated as Zoned(20,0). It is initially set to 1 for each new or restored journal and is incremented until you request that it be reset when you attach a new receiver. There are occasional gaps in the sequence numbers because the system uses internal journal entries for control purposes. These gaps occur if you use commitment control, journal physical files, or journal access paths.

System name. The name of the system on which the entry is being retrieved, if the journal receiver was attached prior to installing V4R2M0 on the system. If the journal receiver was attached while the system was running V4R2M0 or a later release, the system name is the system where the journal entry was actually deposited.

System sequence number. The system sequence number indicates the relative sequence of when this journal entry was deposited into the journal. The system sequence number could be used to sequentially order journal entries that are in separate journal receivers. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*SYSSEQ) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then Hex 0 will be returned for the system sequence number.

Thread identifier. Identifies the thread within the process that added the journal entry. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*THD) was not in effect for the journal when the journal receiver that contains this journal entry was attached, then hex 0 will be returned for the thread identifier.

Time stamp. The system date and time when the journal entry was added to the journal receiver. The time stamp is in the format YYYY-MM-DD-HH.MM.SS.UUUUUU where

YYYY	Year
MM	Month
DD	Day
HH	Hours
MM	Minutes
SS	Seconds
UUUUUU	Microseconds

The system cannot assure that the time stamp is always in ascending order for sequential journal entries because the value of the system time could have been changed.

Note: If the system value QLEAPADJ (Leap year adjustment) is zero, then the result returned will be in 1 microsecond granularity. If the system value QLEAPADJ is greater than zero, then the result returned will be in 8 microsecond granularity.

Transaction identifier. The transaction identifier associated with this journal entry. The transaction identifier identifies transactions related to specific commit cycles. See the QSYSINC/H.XA header file for the layout of this data. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*XID) was not in effect for the journal when the journal receiver that contains the journal entry was attached, then the displacement to transaction identifier will be 0 and no transaction identifier will be returned.

Trigger. Whether this entry was created as result of a trigger program.

0 This entry was not created as the result of a trigger program.
1 This entry was created as the result of a trigger program.

Unformatted time stamp. The system date and time when the journal entry was added to the journal receiver. The time stamp is in machine readable format and can be used as input to a time conversion API, which will convert it to a human readable format.

User name. The user profile name of the user that started the job.

Notes:

1. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*JOB) was not in effect for the journal when the journal receiver that contains the journal entry was attached, then blanks are returned for the user name.
2. If the job name was not available when the journal entry was deposited, then blanks are returned for the user name.

User profile. The name of the effective user profile under which the job was running when the entry was created.

Notes:

1. If RCVSIZOPT(*MINFIXLEN) was in effect or FIXLENDTA(*JOB) was not in effect for the journal when the journal receiver that contains this journal entry was attached, *OMITTED is returned for the effective user profile.
2. If the journal entry was deposited by a system task that was not associated with a job, then a character representation of the task description entry number will be returned for the user profile.

Use of Pointers within Entry Specific Data

There are some journal entries that require additional handling of the journal receiver entry specific data using pointers. This was done to minimize movement of large amounts of data and to facilitate support of tables or database files with large object (LOB) fields. » Here are some examples of journal entries that may require pointer support: «

- Any operations on specific records or files (journal code R or F) of tables or database files that include any fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object). See the SQL programming and DB2® for i5/OS® SQL reference topic collections for more information about these data types.
- Operations related to byte stream file write operations, Journal Code B. See the Integrated file system topic collection for more information about these journal entries.
- Operations related to data queue send operations, Journal Code Q, Entry types QK and QS. See the Journal management topic collection for more information about these journal entries.

- Any operations on specific records or files (journal code R or F) of tables or database files resulting in minimized entry specific data when the journal has MINENTDTA specified for the corresponding object type. See the Journal management topic collection for restrictions and usage of journal entries with minimized entry specific data.
- » User-created entries (journal code U).«

If the incomplete data indicator is returned as a 1, then that indicates that the journal-entry specific data includes a pointer to additional data. Additionally, a pointer handle will be returned with the journal entry. This handle is associated with any allocations required to support the pointer processing.

The pointer must be used by the same process that called this API; it cannot be stored and used by a different process. The pointer can be used for read access only. See the Journal management topic collection for descriptions of the entry types that may include pointer data. The pointer can be used in the following way:

- It can be used directly to copy the data addressed to some other storage space.
- If the journal entry is a record entry (journal code R), the journal-entry specific data could be used for an update or insert operation to the database file through SQL. See the DB2 for i5/OS SQL reference topic collection for more information.

The pointer handles will be implicitly deleted when the process that requested the journal entries is ended.

These pointers can be used only with the V4R4M0 or later versions of the following languages:

- ILE COBOL
- ILE RPG
- ILE C if the TERASPACE parameter is used when compiling the program. See the WebSphere® Development Studio: ILE C/C++ Programmer's Guide manual for more information.

Once the pointer data is used, you must delete the pointer handle to free the handle and any allocations associated with that handle. This can be done by using the "Delete Pointer Handle (QjoDeletePointerHandle) API" on page 34. If the handles are not deleted, the maximum number allowed can be reached, which will prevent further retrieval of journal entries. The deletion must occur from the same process that called the Retrieve Journal Entries (QjoRetrieveJournalEntries) API.

Even if the journal entry data is not used, all pointer handles returned to the user through this interface should be deleted. This is also true when partial journal entry information is returned, even though an error message was returned.

Note: No system function will prevent the deletion of journal receivers that may have outstanding pointer handles. If you want to prevent the journal receivers from being deleted prior to your use of the pointers, you may want to consider using the "Delete Journal Receiver Exit Program" on page 171 exit point, QIBM_QJO_DLT_JRNRCV.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF3C90 E	Literal value cannot be changed.

Message ID	Error Message Text
CPF694A E	Number of fields &1 for key &2 is not valid.
CPF694B E	Length &1 of variable record for key &2 not valid.
CPF694C E	Variable length record data for key &1 not valid.
CPF6946 E	Number &1 specified for key &2 not valid.
CPF6948 E	Length of the receiver variable &1 is not valid.
CPF6949 E	Pointer to a receiver variable is not valid.
CPD700D E	Incorrect FILE, OBJ, OBJFID, OBJPATH, or OBJJID specification.
CPD7025 E	Value &1 for OBJJID not valid.
CPD7061 E	FROMENT and FROMTIME parameters cannot be used together.
CPD7062 E	TOENT and TOTIME parameters cannot be used together.
CPD7076 E	Value specified for JRNCDE not valid.
CPD7078 E	Duplicate journal code not valid.
CPF7002 E	File &1 in library &2 not a physical file.
CPF7006 E	Member &3 not found in file &1 in &2.
CPF7007 E	Cannot allocate member &3 file &1 in &2.
CPF701B E	Journal recovery of interrupted operation failed.
CPF705C E	INCENT(*ALL) not allowed for a local journal.
CPF7053 E	Values for RCVRNG parameter not correct; reason code &1.
CPF7054 E	FROM and TO values not valid.
CPF7055 E	Maximum number of files and members exceeded.
CPF7057 E	*LIBL not allowed with FILE(*ALL).
CPF706A E	Significant null value indicator truncated.
CPF7060 E	Object not found and not journaled in specified receiver range.
CPF7061 E	Conversion of journal entries failed.
CPF7062 E	No entries converted or received from journal &1.
CPF7065 E	Entry type (ENTTYP) not valid for journal code (JRNCDE).
CPF7074 E	RCVRNG for specified SEARCH not valid.
CPF708D E	Journal receiver found logically damaged.
CPF709C E	JOB, PGM, and USRPRF not valid for receiver range.
CPF70A9 E	OBJPATH parameter not valid for a remote journal.
CPF70AC E	FID &1 not found.
CPF70AE E	Member *FIRST not allowed for a remote journal.
CPF8100 E	All CPF81xx messages could be returned. 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.
CPF9809 E	Library &1 cannot be accessed.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9822 E	Not authorized to file &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

Example

The following example retrieves one journal entry based on four keys that will be passed in the Variable Length Record structure.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 180.

```
/*********************************************************************/
/* Setup instructions:                                               */
/*      CRTLIB     RJESAMPLE                                         */
/*      CRTJRNRCV  JRNRCV(RJESAMPLE/R1)                              */
/*      CRTJRN     JRN(RJESAMPLE/J1) JRNRCV(RJESAMPLE/R1)            */
/*      CRTPF      FILE(RJESAMPLE/F1) RCDLEN(12)                     */
```

```
/*      STRJRNPF   FILE(RJESAMPLE/F1) JRN(RJESAMPLE/J1) IMAGES(*BOTH) */
/* Create some journal entries:                                       */
/*      STRSQL                                                        */
/*      INSERT INTO RJESAMPLE/F1 VALUES ('REC1')                      */
/*      INSERT INTO RJESAMPLE/F1 VALUES ('REC2')                      */
/*      INSERT INTO RJESAMPLE/F1 VALUES ('REC3')                      */
/*      DELETE FROM RJESAMPLE/F1 WHERE F1 = 'REC2'                    */
/*      F3 to exit, then ENTER                                        */
/*                                                                    */
/* In this example, we are only going to retrieve one journal entry.  */
/* When you retrieve more than one, you can just increase the size    */
/* of the receiver variable and then work through the data using the  */
/* displacement values returned in the structure.  All of the         */
/* structures used here are based on structures defined or are        */
/* structures defined in QSYSINC/QJOURNAL.                            */
/**********************************************************************/

/* Some include files we will need */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <qusec.h>
#include <qmhsndpm.h>
#include <qjournal.h>

/* Some constants we should define */
#define LIB  "RJESAMPLE "
#define JRN  "J1        "
#define RCV  "R1        "
#define FILE "F1        "
#define SEQ  "00000000000000000014"

/* These are declares for the Variable Length Record structure
   for the keys */
typedef _Packed struct
{
   Qjo_JE_Fmt_Var_Len_Rcrd_t   base_structure;
   Qjo_JE_Data_t               Data[9004];
} Qjo_JE_Fmt_Var_Len_Rcrd_varlen_t;

typedef _Packed struct
{
   Qjo_JE_Jrn_Info_Retrieve_t  base_structure;
   Qjo_JE_Fmt_Var_Len_Rcrd_varlen_t Fmt_Var_Len_Rcrd[4];
} Qjo_JE_Jrn_Info_Retrieve_varlen_t;


/* Function prototypes */
void buildKey1(Qjo_JE_Data_Key_1_t*);
void buildKey2(Qjo_JE_Data_Key_2_t*);
void buildKey4(Qjo_JE_Data_Key_4_t*);
void buildKey6(Qjo_JE_Data_Key_6_t*);
void copyKeysToVLR(Qjo_JE_Data_Key_1_t*, Qjo_JE_Data_Key_2_t*,
                   Qjo_JE_Data_Key_4_t*, Qjo_JE_Data_Key_6_t*,
                   Qjo_JE_Jrn_Info_Retrieve_varlen_t*);
void printEntryInfo(Qjo_RJNE0100_Hdr_t*);
void sendMsg(int, char*);


const short VLRSize = sizeof(Qjo_JE_Fmt_Var_Len_Rcrd_varlen_t);

void main()
{
    /* declare the key structures - we will input keys 1, 2, 4, and 6 */
    Qjo_JE_Data_Key_1_t   key1;
    Qjo_JE_Data_Key_2_t   key2;
```

```
    Qjo_JE_Data_Key_4_t   key4;
    Qjo_JE_Data_Key_6_t   key6;

    /* declare the structure that will hold the keys */
    Qjo_JE_Jrn_Info_Retrieve_varlen_t infoRetrieve;

    /* Misc variables */
    char qualJrnName[20];
    Qus_EC_t *errCode;
    char errorbuffer[17];
    long int lenRcvVar = 2048;

    /* declare the header structure for format RJNE0100 */
    Qjo_RJNE0100_Hdr_t         *rjne0100Hdr;

    /* build the key structures */
    buildKey1(&key1);
    buildKey2(&key2);
    buildKey4(&key4);
    buildKey6(&key6);

    /* Copy the key structures into Format variable length records */
    memset(&(infoRetrieve), 0x00,
           sizeof(Qjo_JE_Jrn_Info_Retrieve_varlen_t));
    infoRetrieve.base_structure.Num_Var_Len_Rcrds = 0;
    copyKeysToVLR(&key1, &key2, &key4, &key6, &infoRetrieve);

    /* Set up the qualified journal name */
    memcpy(qualJrnName, JRN, sizeof(JRN));
    memcpy(qualJrnName+10, LIB, sizeof(LIB));

    /* Tell the error code structure we want data returned to the
       job log */
    errCode = (Qus_EC_t *) errorbuffer;
    errCode->Bytes_Provided = 0;

    /* Allocate the receiver space */
    if((rjne0100Hdr = (Qjo_RJNE0100_Hdr_t *) malloc(lenRcvVar)) != NULL)
    {
        rjne0100Hdr->Bytes_Returned = 0;

        /* Call the API */
        QjoRetrieveJournalEntries(rjne0100Hdr,
                                  &lenRcvVar,
                                  qualJrnName,
                                  "RJNE0100",
                                  &infoRetrieve,
                                  errCode);

        /* Display the entry information returned (send to job log) */
        printEntryInfo(rjne0100Hdr);
        free(rjne0100Hdr);
    }
    /* That's it :) */
}

void buildKey1(Qjo_JE_Data_Key_1_t *key1)
{
    /* Initialize to all blanks */
    memset(key1, ' ', sizeof(Qjo_JE_Data_Key_1_t));

    /* We will use R1 as both the starting and ending receiver */

    /* Do the starting receiver and receiver lib first */
    memcpy(&(key1->Receiver_Range.Starting_Jrn_Rcv_Name),
           RCV, sizeof(Qjo_Jrn_Rcv_Name_t));
    memcpy(&(key1->Receiver_Range.Starting_Jrn_Rcv_Lib_Name),
```

```
           LIB, sizeof(Qjo_Jrn_Rcv_Lib_Name_t));

    /* Then do the ending receiver and receiver lib */
    memcpy(&(key1->Receiver_Range.Ending_Jrn_Rcv_Name),
           RCV, sizeof(Qjo_Jrn_Rcv_Name_t));
    memcpy(&(key1->Receiver_Range.Ending_Jrn_Rcv_Lib_Name),
           LIB, sizeof(Qjo_Jrn_Rcv_Lib_Name_t));
}

void buildKey2(Qjo_JE_Data_Key_2_t *key2)
{
    /* We will look for the sequence number of the delete entry (R DL).
       On a V5R2 system, that is journal sequence number 14 based on
       the instructions above.  Starting seq num is 14.               */

    /* Initialize key2 structure to NULL */
    memset(key2, 0x00, sizeof(Qjo_JE_Data_Key_2_t));
    memcpy(&(key2->Starting_Seq_Num), SEQ, sizeof(Qjo_Seq_Num_t));
}

void buildKey4(Qjo_JE_Data_Key_4_t *key4)
{
    /* We will look for the sequence number of the delete entry (R DL).
       On a V5R2 system, that is journal sequence number 14 based on
       the instructions above.  Ending seq num is 14.                 */

    /* Initialize key4 structure to NULL */
    memset(key4, 0x00, sizeof(Qjo_JE_Data_Key_4_t));
    memcpy(&(key4->Ending_Seq_Num), SEQ, sizeof(Qjo_Seq_Num_t));
}

void buildKey6(Qjo_JE_Data_Key_6_t *key6)
{
    /* Initialize key6 to NULL */
    memset(key6, 0x00, sizeof(Qjo_JE_Data_Key_6_t));

    /* We will only look for one entry - the R DL entry */
    key6->Number_Entries = 1;
}

void copyKeysToVLR(Qjo_JE_Data_Key_1_t *key1, Qjo_JE_Data_Key_2_t *key2,
                   Qjo_JE_Data_Key_4_t *key4, Qjo_JE_Data_Key_6_t *key6,
                   Qjo_JE_Jrn_Info_Retrieve_varlen_t *infoRetrieve)
{
    short i = infoRetrieve->base_structure.Num_Var_Len_Rcrds;

    /* Key 1 copy */
    infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Var_Len_Rcrd
                 = VLRSize;
    infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Key = 1;
    infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Of_Data =
      sizeof(Qjo_JE_Data_Key_1_t);
    memcpy(&(infoRetrieve->Fmt_Var_Len_Rcrd[i].Data),
          key1, sizeof(Qjo_JE_Data_Key_1_t));
   infoRetrieve->base_structure.Num_Var_Len_Rcrds++;
   i++;

   /* Key 2 copy */
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Var_Len_Rcrd
                = VLRSize;
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Key = 2;
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Of_Data =
      sizeof(Qjo_JE_Data_Key_2_t);
   memcpy(&(infoRetrieve->Fmt_Var_Len_Rcrd[i].Data),
          key2, sizeof(Qjo_JE_Data_Key_2_t));
   infoRetrieve->base_structure.Num_Var_Len_Rcrds++;
   i++;
```

```
   /* Key 4 copy */
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Var_Len_Rcrd
                = VLRSize;
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Key = 4;
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Of_Data =
      sizeof(Qjo_JE_Data_Key_4_t);
   memcpy(&(infoRetrieve->Fmt_Var_Len_Rcrd[i].Data),
          key4, sizeof(Qjo_JE_Data_Key_4_t));
   infoRetrieve->base_structure.Num_Var_Len_Rcrds++;
   i++;

   /* Key 6 copy */
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Var_Len_Rcrd
                = VLRSize;
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Key = 6;
   infoRetrieve->Fmt_Var_Len_Rcrd[i].base_structure.Len_Of_Data =
      sizeof(Qjo_JE_Data_Key_6_t);
   memcpy(&(infoRetrieve->Fmt_Var_Len_Rcrd[i].Data),
          key6, sizeof(Qjo_JE_Data_Key_6_t));
   infoRetrieve->base_structure.Num_Var_Len_Rcrds++;
}

void printEntryInfo(Qjo_RJNE0100_Hdr_t *rjne0100Hdr)
{
    char msg[50];
    Qjo_RJNE0100_JE_Hdr_t *entry_ptr;

    /* get a pointer to the entry */
    entry_ptr = (Qjo_RJNE0100_JE_Hdr_t *)((char *)rjne0100Hdr +
       rjne0100Hdr->Offset_First_Jrn_Entry);

    /* Access the data of interest - we will just print the header,
       sequence number, journal code, and entry type to ensure
       we got the R DL entry */
    memset(msg, ' ', sizeof(msg));
    sprintf(msg, "JH:Bytes Rtrnd:%d\n", rjne0100Hdr->Bytes_Returned);
    sendMsg(sizeof(msg), msg);

    memset(msg, ' ',sizeof(msg));
    sprintf(msg, "JH:Dsp to 1st JEH:%d\n",
            rjne0100Hdr->Offset_First_Jrn_Entry);
    sendMsg(sizeof(msg), msg);

    memset(msg, ' ',sizeof(msg));
    sprintf(msg, "JH:Num ent rtrv:%d\n",
             rjne0100Hdr->Number_Entries_Retreived);
    sendMsg(sizeof(msg), msg);

    memset(msg, ' ',sizeof(msg));
    sprintf(msg, "JH:Cont Hndl:%1.1s\n",
             (char *)&rjne0100Hdr->Continuation_Handle);
    sendMsg(sizeof(msg), msg);

    memset(msg, ' ',sizeof(msg));
    sprintf(msg, "Seq #:%-20.20s\n", entry_ptr->Seq_Number);
    sendMsg(sizeof(msg), msg);

    memset(msg, ' ',sizeof(msg));
    sprintf(msg, "Jrn code:%1.1s\n", (char *)&entry_ptr->Jrn_Code);
    sendMsg(sizeof(msg), msg);

    memset(msg, ' ',sizeof(msg));
    sprintf(msg, "Entry type:%-2.2s\n", entry_ptr->Entry_Type);
    sendMsg(sizeof(msg), msg);
}
```

```
void sendMsg(int length, char *message)
{
    char   msgid[8]       = "CPF9897";
    char   path[21]       = "QCPFMSG   *LIBL     ";
    char   msgtype[11]    = "*INFO     ";
    char   callstcken[11] = "*         ";
    int    callstckco     = 1;
    char   msgkey[5]      = "    ";
    Qus_EC_t *errCode;
    char   errorbuffer[512];

    errCode = (Qus_EC_t *) errorbuffer;
    errCode->Bytes_Provided = 0;

    QMHSNDPM(msgid, path, message, length,
           msgtype, callstcken, callstckco, msgkey,
           errCode);
}
```

API introduced: V4R4

Top | "Journal and Commit APIs," on page 1 | APIs by category

Retrieve Journal Identifier Information (QJORJIDI) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Qualified journal name	Input	Char(20)
4	Journal identifier value	Input	Char(10)
5	Format name	Input	Char(8)
6	Error Code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Journal Identifier Information (QJORJIDI) API retrieves the current name and type of the object associated with the specified journal identifier (JID) for the specified journal. A JID is unique; it is assigned to a particular object when journaling is started for the object. The JID associates the journal entries with a particular object. See "Maintaining a JID for a Journaled Object" on page 102 for more information about how the system maintains a JID value for a journaled object.

This API retrieves the object name and type associated with a particular JID if:

- The specified JID is associated with an object that is journaled to the specified journal.
- The specified JID is associated with an object that was journaled to the specified journal, but the object has since been deleted and a change journal operation[1] has not yet attached a new receiver.
- The specified JID is associated with an object that was journaled to the specified journal, but journaling has since been ended for the object and a change journal operation has not yet attached a new receiver. This is true even if the object is currently journaled to a different journal than the one specified.

This API cannot retrieve an object name for a specified JID if:

- The specified JID was never associated with an object journaled to the specified journal.
- The specified JID is associated with an object that was journaled to the specified journal, but the object has since been deleted and a change journal operation has attached a new receiver.
- The specified JID is associated with an object that was journaled to the specified journal, but journaling has since been ended for the object and a change journal operation has attached a new receiver.

Note: The change journal operation can be a user initiated Change Journal (CHGJRN) command or from system change-journal management support. System change-journal management support is activated by a Create Journal (CRTJRN) or Change Journal (CHGJRN) command with the MNGRCV(*SYSTEM) parameter and value.

If an object name or file identifier cannot be retrieved, blanks are returned for the object name, library name, member name, type, and object file identifier.

The JID for the object associated with a particular journal entry is in the fixed-length portion of the journal entry when specifying:

- OUTPUT(*OUTFILE) and OUTFILFMT(*TYPE4 or *TYPE5) on the Display Journal (DSPJRN) command.
- ENTFMT(*TYPE4 or *TYPEPTR) or ENTFMT(*JRNENTFMT) and JRNENTFMT(RJNE0200) on the Receive Journal Entry (RCVJRNE) command.
- ENTFMT(*TYPE4 or *TYPE5) on the Retrieve Journal Entry (RTVJRNE) command.

Maintaining a JID for a Journaled Object

The following are the system rules for maintaining a JID for a journaled object:

- A JID is assigned for a journaled object when journaling is first started for the object. For example, when journaling is started for a database file with a single member, a JID value is assigned to the data portion of the member. If two members exist in the file, a different JID is assigned to the data portion for each member.
- The JID remains the same for the journaled object if:
  - The object is moved to a different library.
  - The object is renamed.
  - Journaling is ended for the object, and then started again to either the same journal or to a different journal.
  - The object is saved while being journaled, deleted from the system, and then restored from the saved version. This is true whether the object is being restored to the original system from which it was saved or to a different system, as long as the same version of the object does not already exist on that system.
- The JID does not remain the same for the journaled object if:
  - The object is deleted, and then created again. When journaling is started for the newly created object, a new JID is assigned. The object retains its original JID when the Replay Database Operation (QDBRPLAY) API or the Replay Journal Entry (QjoReplayJournalEntry) API is used to recreate the object.
  - The object is saved while being journaled, then renamed or moved to another library, and then the object is restored using the saved version. The restored version will be assigned a new JID because the original JID for the object is currently assigned to the original object that was renamed or moved to a different library.

See the Journal management topic for more information about what object types are associated with the various types of journal entries.

Restrictions

- The specified journal cannot be a remote journal.

Authorities and Locks

Journal Authority
*USE, *OBJEXIST

Journal Library Authority
*EXECUTE

Currently Attached Receiver Authority
*USE

Currently Attached Receiver Library Authority
*EXECUTE

Journal Lock
*SHRRD

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

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

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable. The length must be at least 8 bytes. If the variable is not long enough to hold the information, the data is truncated. If the length is larger than the size of the receiver variable, the results beyond the length of the largest format are not predictable.

Qualified journal name
INPUT; CHAR(20)

The name of the journal that is to be used when retrieving the JID information and the library in which it resides. The first 10 characters contain the journal name and the second 10 characters contain the library name. The special values supported for the library name are:

**LIBL*	Library list
**CURLIB*	Current library

Journal identifier (JID) value
INPUT; CHAR(10)

The journal identifier (JID) value that is to be used for the retrieve operation. Information will be retrieved based on this JID value.

Format name
INPUT; CHAR(8)

The format name RJID0100 is the only valid format name used bythis API. For more information, see “RJID0100 Format” on page 104.

Error code
I/O; CHAR(*)

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

RJID0100 Format

The structure of the information returned is determined by the specified format name. For detailed descriptions of the fields, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Journal name
18	12	CHAR(10)	Journal library name
28	1C	CHAR(10)	Journal identifier (JID) value
38	26	CHAR(10)	Object name
48	30	CHAR(10)	Object library name
58	3A	CHAR(10)	Member name
68	44	CHAR(10)	Object type
78	4E	CHAR(16)	Object file identifier

Field Descriptions

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

Bytes returned. The number of bytes of data returned.

Journal identifier (JID) value. The JID value used to retrieve the object name, object library name, member name, and object type.

Journal library name. The name of the library for the journal. If *LIBL or *CURLIB was specified as input, then this field will contain the actual library name for the journal.

Journal name. The name of the journal.

Member name. If the object type is *QDDS or *QDDSI, then this field contains the member name. Otherwise, this field is returned as blanks.

Object file identifier. The file identifier of the object associated with the specified JID value. If the object name could not be retrieved for the specified JID value, then this field is returned as blanks. File identifiers are unique identifiers associated with integrated file system related objects. The Get Path Name of Object from Its File ID (Qp0lGetPathFromFileID) API can be used to find the path name of an object using the file identifier.

If the object type is not *DIR, *STMF, or *SYMLNK, then this field is returned as blanks.

Object library name. The name of the library for the object associated with the specified JID value. If the object name could not be retrieved for the specified JID value, then this field is returned as blanks.

If the object type is *DIR, *STMF, or *SYMLNK, then this field is returned as blanks.

Object name. The name of the object associated with the specified JID value. If the object name could not be retrieved for the specified JID value, then this field is returned as blanks.

If the object type is *DIR, *STMF, or *SYMLNK, then this field is returned as blanks.

Object type. The type of the object associated with the specified JID value. If the object name could not be retrieved for the specified JID value, then this field is returned as blanks. The following lists the valid object types that can be retrieved for a specified JID value:

**DIR*	Integrated file system directory
**DTAARA*	Data area
**DTAQ*	Data queue
**FILE*	Data base file
**JRNRCV*	Journal receiver
**LIB*	Library
**QDDS*	Data portion of a database member
**QDDSI*	Access path for a database member
**STMF*	Integrated file system stream file
**SYMLNK*	Integrated file system symbolic link

Error Messages

Message ID	Error Message Text
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.
CPF3C90 E	Literal value cannot be changed.
CPF6958 E	No attached receiver can be used.
CPF701B E	Journal recovery of interrupted operation failed.
CPF705A E	Operation failed due to remote journal.
CPF708D E	Journal receiver found logically damaged.
CPF8100 E	All CPF81xx messages could be returned. 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.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9825 E	Not authorized to device &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

Top | "Journal and Commit APIs," on page 1 | APIs by category

Retrieve Journal Information (QjoRetrieveJournalInformation) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Qualified journal name	Input	Char(20)
4	Format name	Input	Char(8)
5	Journal information to retrieve	Input	Char(*)

Omissible Parameter:

6	Error Code	I/O	Char(*)

Service Program Name: QJOURNAL
Header File: QSYSINC/H.QJOURNAL
Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Journal Information (QjoRetrieveJournalInformation) API provides access to journal-related information to help manage a journal environment, including a remote journal environment.

Various types of journal information are provided by the API. General information, similar to information reported by using the Work with Journal Attributes (WRKJRNA) CL command, and additional information are contained in the header section. If requested, information is provided for the journal receiver directory, journaled objects, and remote journals.

Authorities and Locks

Journal Authority
*OBJOPR and some data authority other than *EXECUTE

Journal Library Authority
*EXECUTE

ASP Device Authority
*USE

Journal Lock
*SHRRD

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

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

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable 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 if format RJRN0100 is specified. The minimum length is 1 byte if format RJRN0200 is specified.

Note: The format name determines what units are used for the fields **Length of receiver variable**, **Bytes returned**, and **Bytes available**. If format RJRN0100 is specified, the information is in one byte units. If format RJRN0200 is specified, the information is in 4K (4096 bytes) units.

Qualified journal name
INPUT; CHAR(20)

The name of the journal and its library from which the journal attributes and information are to be retrieved. The first 10 characters contain the journal name, and the second 10 characters contain the library name. The special values supported for the library name follow:

*LIBL	Library list
*CURLIB	Current library

Format name
INPUT; CHAR(8)

Format RJRN0100 and RJRN0200 are the only supported formats that are used by this API. For more information, see "RJRN0100/RJRN0200 format" on page 109.

Journal information to retrieve
INPUT; CHAR(*)

Information to be retrieved that is associated with the journal. 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. If this field is zero, no variable length records are processed, and no key information will be retrieved.

Variable length records
CHAR(*)

The types of information that should be retrieved. For the specific format of the variable length record, see "Format for Variable Length Record."

Omissible 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.

Format for Variable Length Record

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

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of variable length record
4	4	BINARY(4)	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 will be truncated at the right.

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

Data. The data that is used to determine how the journal information should be retrieved. All values are validity checked.

Key. Identifies specific information to be retrieved about the journal. See "Keys" for the list of valid keys.

Length of data. The length of the key information.

Length of variable length record. The length of the variable length record. This field is used to get the addressability of the next variable length record.

Keys

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

Key	Input Type	Field
1	N/A	Journal receiver directory information
2	CHAR(10)	Journaled object information
3	CHAR(38)	Remote journal information
» 4	CHAR(10)	Auxiliary storage pool device name «

Note: If you are only interested in summary information about journaled objects, this information is available with format RJRN0100 or format RJRN0200. Key 2 provides detailed information about journaled objects.

Note: The number of objects journaled to the journal can be significant. The time to return the journaled object information (key 2) can also be significant. The space needed to return this information can be more than 2 gigabytes (2,147,483,647).

To get an approximation of the space needed to return the journaled object information (key 2), specify a value of -1 in the input variable **Length of receiver variable**. By doing this, the API will return in the **Bytes available** field the calculated size of space that is needed to return the journaled object information without attempting on this request to get any journaled object information. Since only the calculations will be done, the API will be able to return the requested information quickly. The value of **Bytes available** returned will be in 4K units if format RJRN0200 was specified.

Note: At least 8 bytes must be available in the **Receiver variable** return area to return the **Bytes available** information. Only these 8 bytes of information will be returned when the **Length of receiver variable** is set to -1.

Field Descriptions

» Auxiliary storage pool device name. The name of the auxiliary storage pool (ASP) in which the journal resides. This value may be specified for an independent ASP that is in ACTIVE or AVAILABLE status. If this key input is omitted in cases where it is valid for this key to have a value other than an asterisk (*), the thread's library name space will be used. The possible values follow:

*	The ASPs in the thread's library name space.
ASP device name	The name of the independent ASP where the journal resides. Valid names depend on the ASPs active or available on the system.

«

Journaled object information. The list of objects that are journaled to the specified journal of specific object types. The input key value indicates what journaled object information to retrieve. The object types that are supported for retrieval are *FILE, *DTAARA, *DTAQ, »*LIB«, *DIR, *STMF, and *SYMLNK. The possible values follow:

**ALL*
: All objects that are journaled to the journal, of the object types supported for retrieval with this key, are returned.

**ALLIFS*
: Only the following object types that are journaled to the journal are returned: *DIR, *STMF, and *SYMLNK.

Object type
: Only the objects of the specified object type that are journaled to the journal are returned.

  Note: The following object types cannot be specified individually in this field: *DIR, *STMF, and *SYMLNK. To retrieve these object types, specify *ALLIFS.

For output values, see "Key 2 Output Section" on page 112.

Journal receiver directory information. The journal receiver directory information that is returned for the journal. Journal receiver directory information can be retrieved for local and remote journals. There are no input values for this key. For output values, see "Key 1 Output Section" on page 111.

Remote journal information. The remote journal information that is returned. Remote journal information can be retrieved for local and remote journals. The input key value indicates what remote journal information to retrieve. The possible values follow:

Char(18)
: The relational database directory entry information as follows:

  **ALL*
  : All remote journal information is returned.

  Relational database directory entry
  : Only the remote journal information for the specified relational database directory entry is to be returned.

Char(20)
: The remote journal name information as follows:

  **ALL*
  : All remote journals that are associated with the specified relational database directory entry are returned.

Qualified remote journal name
: Only the remote journal information for the specified journal that is associated with the specified relational database directory entry is to be returned. The first 10 characters contain the remote journal name, and the second 10 characters contain the library name.

For output values, see "Key 3 Output Section" on page 112.

RJRN0100/RJRN0200 format

The structure of the information returned is determined by the specified format name. For detailed descriptions of the fields, see "Field Descriptions" on page 114.

Note: The format name determines what units are used for the fields **Bytes returned** and **Bytes available**. If format RJRN0100 is specified, the information is in one byte units. If format RJRN0200 is specified, the information is in 4K (4096 bytes) units.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Offset to key information
12	C	CHAR(10)	Journal name

Offset		Type	Field
Dec	Hex		
22	16	CHAR(10)	Journal library name
32	20	BINARY(4)	Auxiliary storage pool (ASP)
36	24	CHAR(10)	Message queue name
46	2E	CHAR(10)	Message queue library name
56	38	CHAR(1)	Manage receiver option
57	39	CHAR(1)	Delete receiver option
58	3A	CHAR(1)	Receiver size option *RMVINTENT
59	3B	CHAR(1)	Receiver size option *MINFIXLEN
60	3C	CHAR(1)	Receiver size option *MAXOPT1
61	3D	CHAR(1)	Receiver size option *MAXOPT2
62	3E	CHAR(1)	Receiver size option *MAXOPT3
63	3F	CHAR(2)	Reserved
65	41	CHAR(1)	Journal type
66	42	CHAR(1)	Remote journal type
67	43	CHAR(1)	Journal state
68	44	CHAR(1)	Journal delivery mode
69	45	CHAR(10)	Local journal name
79	4F	CHAR(10)	Local journal library name
89	59	CHAR(8)	Local journal system
97	61	CHAR(10)	Source journal name
107	6B	CHAR(10)	Source journal library name
117	75	CHAR(8)	Source journal system
125	7D	CHAR(10)	Redirected receiver library name
135	87	CHAR(50)	Journal text
185	B9	CHAR(1)	Minimize entry specific data for data areas
186	BA	CHAR(1)	Minimize entry specific data for files
187	BB	CHAR(8)	Reserved
195	C3	CHAR(1)	Journal Cache
196	C4	BINARY(4)	Number of attached journal receivers
200	C8	CHAR(10)	Attached journal receiver name
210	D2	CHAR(10)	Attached journal receiver library name
220	DC	CHAR(8)	Local journal system associated with the attached journal receiver
228	E4	CHAR(8)	Source journal system associated with the attached journal receiver
236	EC	CHAR(10)	Attached dual journal receiver name
246	F6	CHAR(10)	Attached dual journal receiver library name
256	100	BINARY(4)	Manage receiver delay
260	104	BINARY(4)	Delete receiver delay
264	108	CHAR(10)	ASP device name
274	112	CHAR(10)	Local journal ASP group name
284	11C	CHAR(10)	Source journal ASP group name

Offset		Type	Field
Dec	Hex		
294	126	CHAR(1)	Fixed length data JOB
295	127	CHAR(1)	Fixed length data USR
296	128	CHAR(1)	Fixed length data PGM
297	129	CHAR(1)	Fixed length data PGMLIB
298	12A	CHAR(1)	Fixed length data SYSSEQ
299	12B	CHAR(1)	Fixed length data RMTADR
300	12C	CHAR(1)	Fixed length data THD
301	12D	CHAR(1)	Fixed length data LUW
302	12E	CHAR(1)	Fixed length data XID
303	12F	CHAR(4)	Reserved
307	133	CHAR(1)	Journaled object limit
308	134	BINARY(4)	Total number of journaled objects
312	138	BINARY(4)	Total number of journaled files
316	13C	BINARY(4)	Total number of journaled members
320	140	BINARY(4)	Total number of journaled data areas
324	144	BINARY(4)	Total number of journaled data queues
328	148	BINARY(4)	Total number of journaled integrated file system objects of type *DIR, *STMF, and *SYMLNK
332	14C	BINARY(4)	Total number of journaled access paths
336	150	BINARY(4)	Total number of commitment definitions
340	154	BINARY(4)	Journal recovery count
» 344	158	BINARY(4)	Total number of journaled libraries
348	15C	CHAR(100) «	Reserved
448	1C0	BINARY(4)	Number of keys in key section
Note: These fields repeat for each key specified.			
		BINARY(4)	Key
		BINARY(4)	Offset to start of key information
		BINARY(4)	Length of key information header section
		BINARY(4)	Number of entries
		BINARY(4)	Length of each entry in key information list section

Key 1 Output Section

Offset		Type	Field
Dec	Hex		
Note: The following fields are returned when the journal receiver directory key information is specified. Otherwise, they will not be used.			
0	0	BINARY(4)	Total number of journal receivers
4	4	BINARY(4)	Total size of journal receivers
8	8	BINARY(4)	Total size of journal receivers multiplier

Offset		Type	Field
Dec	Hex		
12	C	CHAR(8)	Reserved
Note: The following fields repeat for each journal receiver that is returned.			
		CHAR(10)	Journal receiver name
		CHAR(10)	Journal receiver library name
		CHAR(5)	Journal receiver number
		CHAR(13)	Journal receiver attached date and time
		CHAR(1)	Journal receiver status
		CHAR(13)	Journal receiver saved date and time
		CHAR(8)	Local journal system associated with the journal receiver
		CHAR(8)	Source journal system associated with the journal receiver
		BINARY(4)	Journal receiver size
		CHAR(56)	Reserved

Key 2 Output Section

Offset		Type	Field
Dec	Hex		
Note: The following fields are returned when the journaled object information key is specified. Otherwise, they will not be used.			
0	0	BINARY(4)	Total number of journaled files
4	4	BINARY(4)	Total number of journaled members
8	8	BINARY(4)	Total number of journaled data areas
12	C	BINARY(4)	Total number of journaled data queues
16	10	BINARY(4)	Total number of journaled integrated file system objects of type *DIR, *STMF, and *SYMLNK
» 20	14	BINARY(4)	Total number of journaled libraries
24	18	CHAR(12) «	Reserved
Note: The following fields repeat for each journaled object that is returned.			
		CHAR(10)	Object type
		CHAR(10)	Object name
		CHAR(10)	Object library name
		CHAR(16)	Object file identifier
		» CHAR(1)	File type «
		» CHAR(1) «	Reserved

Key 3 Output Section

Offset		Type	Field
Dec	Hex		
Note: The following fields are returned when the remote journal information key is specified. Otherwise, they will not be used.			

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Total number of remote journals
4	4	CHAR(16)	Reserved
Note: The following fields repeat for each remote journal that is returned.			
		CHAR(18)	Relational database directory entry
		CHAR(10)	Remote journal name
		CHAR(10)	Remote journal library name
		CHAR(10)	Remote journal receiver library name
		CHAR(10)	Controlled-inactivate journal information journal receiver
		CHAR(10)	Controlled-inactivate journal information journal receiver library
		BINARY(4)	Controlled-inactivate journal information sequence number
		BINARY(4)	Reserved
		CHAR(1)	Remote journal type
		CHAR(1)	Remote journal state
		CHAR(1)	Remote journal delivery mode
»		CHAR(1)	Validity checking «
		BINARY(4)	Sending task priority
		CHAR(20)	Controlled-inactivate journal information sequence number - long
»		BINARY(4)	Synchronous sending time out
		BINARY(4)	Number of entries behind
		BINARY(4)	Maximum entries behind
		CHAR(13)	Maximum entries behind time
		CHAR(3)	Reserved
		BINARY(4)	Hundredths of seconds behind
		BINARY(4)	Maximum hundredths of seconds behind
		CHAR(13)	Maximum hundredths of seconds behind time
		CHAR(11) «	Reserved
		CHAR(512)	Relational database directory entry details
»		CHAR(8)	Node identifier
		BINARY(4)	Offset to internet address array
		BINARY(4)	Number of internet addresses
		BINARY(4)	Number of active internet addresses
		CHAR(13)	Number of active internet addresses time
		CHAR(13)	Remote journal last catch-up time

Offset		Type	Field
Dec	Hex		
		CHAR(13)	Remote journal active state time
		CHAR(1)	Reserved
		BINARY(4)	Number of bundles
		BINARY(4)	Maximum bundle size
		CHAR(13)	Maximum bundle size time
		CHAR(3)	Reserved
		BINARY(4)	Super bundle count
		CHAR(80)	Reserved
		Array(*) of CHAR(45)	Internet address «

Field Descriptions

Attached dual journal receiver library name. The name of the library that contains the dual journal receiver.

This field is blank if there is no dual receiver.

Attached dual journal receiver name. The journal receiver that was attached at the same time as the attached journal receiver.

This field is blank if there is no dual receiver.

Attached journal receiver library name. The name of the library that contains the attached journal receiver. This field will be blank if no journal receivers are attached.

Attached journal receiver name. The name of the journal receiver that is currently attached to this journal. This field will be blank if no journal receivers are attached.

Auxiliary storage pool (ASP). The number of the auxiliary storage pool to which storage for the object is allocated.

ASP device name. The name of the independent auxiliary storage pool (ASP) to which storage for the object is allocated. *SYSBAS is used to indicate the system ASP and all basic user ASPs.

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

Note: The format name determines what units are used for this field. If format RJRN0100 is specified, the information is in one byte units. If format RJRN0200 is specified, the information is in 4K (4096 bytes) units.

Bytes returned. The number of bytes of data returned.

Note: The format name determines what units are used for this field. If format RJRN0100 is specified, the information is in one byte units. If format RJRN0200 is specified, the information is in 4K (4096 bytes) units.

Controlled-inactivate journal information journal receiver. The name of the journal receiver that contains the controlled inactivate journal information sequence number.

This field will be blank unless the remote journal state is *CTLINACT.

Controlled-inactivate journal information journal receiver library. The library of the journal receiver that contains the controlled inactivate journal information sequence number.

This field will be blank unless the remote journal state is *CTLINACT.

Controlled-inactivate journal information sequence number. The sequence number of the last journal entry that was queued for replication before the Change Remote Journal (CHGRMTJRN) command or the Change Journal State (QjoChangeJournalState) API was called to start a controlled inactivate of the remote journal.

This field will be 0 unless the remote journal state is *CTLINACT.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will be in the Controlled-inactivate journal information sequence number - long field.

Controlled-inactivate journal information sequence number - long. This is the same field as Controlled-inactivate journal information sequence number except the information is in a Char(20) field which is treated as Zoned(20,0).

Delete receiver delay. The delay time (in minutes) between attempts to delete journal receivers associated with this journal if the delete receiver option is a 1. The default is 10 minutes.

Delete receiver option. Whether the system deletes detached journal receivers that are associated with this journal when they are no longer needed for IPL recovery.

0 The system does not delete detached journal receivers that are associated with this journal.
1 The system deletes detached journal receivers that are associated with this journal.

File type. The type of file journaled.

blank Blank is returned when the object is not a file.
0 Journaled file is a physical file type.
1 Journaled file is a logical file type.

Fixed length data *JOB. Indicates whether the job name will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the job name.
1 Journal entries deposited to the journal will include the job name.

Fixed length data *LUW. Indicates whether the logical unit of work identifier will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the logical unit of work identifier.
1 Journal entries deposited to the journal may include the logical unit of work identifier.

Fixed length data *PGM. Indicates whether the program name will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the program name.
1 Journal entries deposited to the journal will include the program name.

Fixed length data *PGMLIB. Indicates whether the program library name will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the program library name and library ASP information.
1 Journal entries deposited to the journal will include the program library name and library ASP information.

Fixed length data *RMTADR. Indicates whether the remote address will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the remote address.
1 Journal entries deposited to the journal may include the remote address.

Fixed length data *SYSSEQ. Indicates whether the system sequence number will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the system sequence number.
1 Journal entries deposited to the journal will include the system sequence number.

Fixed length data *THD. Indicates whether the thread identifier will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the thread identifier.
1 Journal entries deposited to the journal will include the thread identifier.

Fixed length data *USR. Indicates whether the user name will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the user name.
1 Journal entries deposited to the journal will include the user name.

Fixed length data *XID. Indicates whether the transaction identifier will be stored when journal entries are deposited.

blank Blank is returned when the journal is a remote journal.
0 Journal entries deposited to the journal will not include the transaction identifier.
1 Journal entries deposited to the journal may include the transaction identifier.

Hundredths of seconds behind. If the remote journal delivery mode is *ASYNC, this is the hundredths of seconds that the source journal is behind in sending journal entries to the target system.

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field is -1. «

» **Internet address.** Up to four internet addresses being used by data port services to communicate to the target system. If a node identifier and at least one internet address is retrieved, then data port services is being used as an alternate communication method to the target system.

Note: The offset to the internet address array for each environment is determined by the **Offset to internet address array** field.

Note: The internet address array will have the same length for each remote journal environment returned. The internet address array length will be large enough to hold the maximum number of internet addresses that can be specified for data port services. The internet address array will be populated with internet addresses up to the **Number of internet addresses** returned. Any remaining internet addresses in the array will be blank. «

Journal cache. Specifies whether journal entries were cached before being written out to disk.

blank Blank is returned when the journal is a remote journal.

0 Journal entries are written to disk immediately if needed to assure single-system recovery.

1 Journal entries are written to main memory. When there are several journal entries in main memory then the journal entries are written from main memory to disk. If the application performs large numbers of changes, this may result in fewer synchronous disk writes resulting in improved performance. However, is is **not recommended** to use this option if it is unacceptable to lose even one recent change in the event of a system failure where the contents of main memory are not preserved. This type of journaling is directed primarily toward batch jobs and may not be suitable for interactive applications where single system recovery is the primary reason for using journaling.

Note: Applications using commitment control will likely see less performance improvement because commitment control already performs some journal caching.

Journal delivery mode. The journal delivery mode that is being used to replicate journal entries to this journal.

0 Not applicable. This is a local journal or this remote journal is not *ACTIVE or not *CTLINACT.

1 *ASYNC. Journal entries are being delivered or replicated asynchronously.

2 *SYNC. Journal entries are being delivered or replicated synchronously.

3 *ASYNCPEND. Journal entries are to be delivered or replicated asynchronously, but the journal is currently in catch-up mode.

4 *SYNCPEND. Journal entries are to be delivered or replicated synchronously, but the journal is currently in catch-up mode.

Journal library name. The name of the library that contains the journal.

Journal name. The name of the journal.

Journal receiver attached date and time. The date and time that this journal receiver was attached to the journal. For a journal receiver attached to a *REMOTE journal, this is the date and time that the journal receiver was attached on the local system. This field is in the CYYMMDDHHMMSS format as follows:

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

Journal receiver library name. The name of the library that contains the journal receiver.

Journal receiver name. The name of the journal receiver.

Journal receiver number. A number that is associated with a journal receiver and assigned by the system, which is relative to all other journal receivers in the journal receiver directory at this time. For a given journal receiver, this number will change as journal receivers are added and deleted from the directory. The first 2 digits identify the journal chain number and the last 3 digits identify the journal receiver number within the chain.

A chain identifies a group of journal receivers that are contiguous, which allows the system to process entries across journal receivers within the same chain.

The chain number starts with zero and is incremented sequentially each time a new chain is needed. For example, new chains are started when a damaged journal receiver is recovered by restoring a partial version.

Within a chain, each newly attached journal receiver is given a journal receiver number starting with one and incrementing sequentially to 999.

When you journal to dual journal receivers, both journal receivers are assigned the same number.

Journal receiver saved date and time. The date and time that the journal receiver was last saved. This field is in the CYYMMDDHHMMSS format, which is described in the journal receiver attached date and time field description.

Journal receiver size. The number of kilobytes of auxiliary disk storage used by this journal receiver.

This field will be zero if the journal receiver is damaged.

Journal receiver status. The status of the journal receiver. The status can be one of the following:

1 The journal receiver is currently attached to the journal.

2 The journal receiver is online. The journal receiver has not been saved, and it has been detached from the journal.

3 The journal receiver was saved after it was detached. The journal receiver storage was not freed when it was saved.

4 The journal receiver was saved after it was detached. The journal receiver storage was freed when it was saved.

5 The journal receiver status is partial for one of the following reasons:

- It was restored from a version that was saved while it was attached to the journal. Additional journal entries may have been written that were not restored.
- It was one of a pair of dual journal receivers, and it was found damaged while attached to the journal. The journal receiver has since been detached. This journal receiver is considered partial because additional journal entries may have been written to the dual journal receiver.
- It is associated with a remote journal and it does not contain all the journal entries that are in the corresponding journal receiver associated with the source journal.

Journal recovery count. The journal recovery count allows a user to choose between faster abnormal IPL or independent ASP vary on recovery and decreased run time processing. The value specified influences the frequency with which journaled objects are forced to auxiliary storage as those objects are changed. The specified journal recovery count indicates the approximate number of journaled changes that would

need to be recovered during journal synchronization for this journal in the event of an abnormal IPL or vary on. Specifying a smaller value decreases the number of changes that would need to be recovered from this journal in the event of an abnormal IPL or vary on by increasing the frequency with which changed objects are forced. Specifying a larger value increases the number of changes that would need to be recovered for this journal during an abnormal IPL or vary on by decreasing the frequency with which changed objects are forced. Changing this value may affect overall system performance as it affects the utilization of auxiliary storage devices.

All journals are created with the system default journal recovery count. If a value other than the system default is specified, the system default journal recovery count will no longer be in effect for this journal.

The operating system is shipped with a system default journal recovery count of 250,000. If there is a need to change the system default journal recovery count for all newly created journals and all existing journals that have the system default (*SYSDFT) specified for their journal recovery count, please refer to the Change Journal Recovery Count (QJOCHRVC) API.

0 The value is set to the system default journal recovery count.
10000-2000000000 Specifies the approximate number of journal entries that may need to be recovered from this journal during an abnormal IPL or vary on.

Journal state. An indication as to whether journal entries are currently being sent to a journal. For a remote journal, this is whether the journal is actively receiving journal entries from the source system journal.

0 *INACTIVE. If this is a remote journal, this means journal entries cannot be received from a source journal.
1 *ACTIVE. If this is a local journal, this means journal entries can be deposited to this journal. If this is a remote journal, this means journal entries can be received from a source journal.
2 *FAILED. If this is a remote journal, this means journal entries cannot be received from a source journal due to a remote journal function failure, for example, a communications failure. Before inactivating the remote journal by using the Change Remote Journal (CHGRMTJRN) command or by calling the Change Journal State (QjoChangeJournalState) API, you may want to receive, retrieve, or display any unconfirmed entries from this journal.

This value does not apply to local journals.
4 *INACTPEND. If this is a remote journal, this means a request is being processed to set the journal state to *INACTIVE. Or, the remote journal was a target in a synchronous environment and the environment has gone down, leaving unconfirmed entries in the journal.

This value does not apply to local journals.
5 *STANDBY. If this is a local journal, this means that most journal entries are not deposited into the journal and there will be no errors indicating that the entry was not deposited. While in standby state, journaling can be started or stopped, however using explicit commitment control is not allowed. Also, databases files that have referential integrity constraints or data links defined cannot be used when a journal is in standby state. The operating system needs to use commitment control for these functions. However, referential integrity constraints can be used in standby state if RESTRICT is specified on the ON UPDATE or ON DELETE attribute for the constraint.

This value applies only to local journals.

Journal text. The text description of the journal.

Journal type. The journal type defines the scope of the journal and some of its characteristics. The valid journal types follow:

0 *LOCAL

1 *REMOTE

Journaled object limit. The journaled object limit defines how many objects can be journaled to the journal. The valid journaled object limits follow:

blank	Blank is returned when the journal is a remote journal.
0	up to 250,000 objects
1	up to 10,000,000 objects

Key. Specific information retrieved about the journal.

Length of key information header section. The length of the header information in the given keys information section. The header is followed by the list information section, which is a repeating list of entries for the given key.

Length of each entry in key information list section. The length of an entry within a specific list that is returned for a given key.

Local journal ASP group name. The name of the independent auxiliary storage pool (ASP) group of the local journal. *SYSBAS is used to indicate the system ASP and all basic user ASPs. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

Local journal library name. The library name of the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

Local journal name. The journal name of the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

Local journal system. The system name of the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This system name is determined when the remote journal is activated by using the Change Remote Journal (CHGRMTJRN) command or by calling the Change Journal State (QjoChangeJournalState) API. The name is based on the current system name at that time as seen by using the Display Network Attributes (DSPNETA) command.

This field is blank if there is no local journal.

This field is *UNKNOWN if no journal receiver is currently attached or if the journal receiver is damaged or destroyed.

Local journal system associated with the attached journal receiver. If this attached journal receiver is associated with a remote journal, this field is the system name of the local journal. See the local journal system field for more information.

Local journal system associated with the journal receiver. If this journal receiver was associated with a remote journal, this field is the system name of the local journal. See the local journal system field for more information.

Manage receiver delay. The delay time (in minutes) between attempts to attach new journal receivers to this journal if the manage receiver option is a value of 1. The default is 10 minutes.

Zero is returned when the journal is a remote journal.

Manage receiver option. Whether the system or user manages the changing of journal receivers; that is, detaching the currently attached journal receivers and attaching new journal receivers. This option is applicable only for local journals and is blank for remote journals. Possible values follow:

blank	Blank is returned when the journal is a remote journal.
0	The user manages the changing of journal receivers by issuing the Change Journal (CHGJRN) command to attach new journal receivers and detach old journal receivers.
1	The system manages the changing of journal receivers. When an attached journal receiver reaches its size threshold, the system creates and attaches new journal receivers, and detaches the currently attached journal receivers. Additionally, during an initial program load (IPL), the system performs a Change Journal (CHGJRN) command to change journal receivers and reset the journal sequence number if the journal is not needed to complete commitment-control IPL recovery.

Maximum bundle size. The number of bytes in the largest bundle that has been sent to the target system since the source journal transitioned to an active state.

If the remote journal state is *INACTIVE, this field is -1.

Maximum bundle size time. The date and time that the maximum bundle size was sent to the target system. This field is in the CYYMMDDHHMMSS format as follows:

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

If the remote journal state is *INACTIVE, this field will be blank.

Maximum entries behind. If the remote deliver mode is *ASYNC, this is the maximum number of entries that were waiting to be sent to the target system.

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field is -1.

Maximum entries behind time. If the remote deliver mode is *ASYNC, this is the date and time that the maximum entries behind occurred. This field is in the CYYMMDDHHMMSS format as follows:

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

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field will be blank.

Maximum hundredths of seconds behind. If the remote deliver mode is *ASYNC, this is the maximum hundredths of seconds that the source journal was behind in sending journal entries to the target system.

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field is -1.

Maximum hundredths of seconds behind time. If the remote deliver mode is *ASYNC, this is the date and time that the maximum hundredths of seconds behind occurred. This field is in the CYYMMDDHHMMSS format as follows:

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

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field will be blank.

Message queue library name. The name of the library that contains the message queue.

Message queue name. The name of the message queue that is associated with this journal. This message queue will receive various messages that describe the operations on the journal. For example, if the threshold value of the attached journal receiver is exceeded during journaling and the journal currently is being managed by the user, a CPF7099 message is sent to this message queue. If the journal is being managed by the system, then CPF7020 is sent to this message queue when the change journal has successfully completed. Messages issued by the remote journal support will also be sent to this message queue. The MNGRCV parameter on the Create Journal (CRTJRN) command or on the Change Journal (CHGJRN) command) specifies whether the journal is being managed by the user or by the system.

Minimize entry specific data for data areas. Whether journal entries for data areas may have minimized entry specific data. The possible values are:

blank Blank is returned when the journal is a remote journal.
0 Journal entries for data areas will have complete entry specific data.
1 Journal entries for data areas may have minimized entry specific data.

Minimize entry specific data for files. Whether journal entries for files may have minimized entry specific data. The possible values are:

blank Blank is returned when the journal is a remote journal.
0 Journal entries for files will have complete entry specific data.
1 Journal entries for files may have minimized entry specific data. The minimizing does not occur on field boundaries. Therefore, the entry specific data may not be viewable and may not be used for auditing purposes.
2 Journal entries for files may have minimized entry specific data. The minimizing occurs on field boundaries. Therefore, the entry specific data will be viewable and may be used for auditing purposes.

» **Node identifier.** The node identifier being used by data port services to identify the target system in a cluster environment. If a node identifier and at least one internet address is retrieved, then data port services is being used as an alternate communication method to the target system.

This field will be set to *NONE if the remote journal is not configured for data port services. If this field is *NONE, the relational database is used for all communications to the target system.

Number of active internet addresses. The number of internet addresses that are active for data port services.

This field will be -1 when the remote journal is not configured for data port services.

Number of active internet addresses time. The date and time that the number of active internet addresses last changed. This field is in the CYYMMDDHHMMSS format as follows:

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

This field will be blank when the remote journal is not configured for data port services. «

Number of attached journal receivers. The number of journal receivers that are currently attached to the journal. If this number is two, the dual journal receiver information is returned. If this number is one or zero, the dual journal receiver information is returned as blanks. If this number is zero, the journal receiver information is returned as blanks, and if the journal receiver directory key is specified, there will be no entries returned.

» **Number of bundles.** This is the number of bundles that have been sent to the target system since the source journal transitioned to an active state.

If the remote journal state is *INACTIVE, this field is -1. «

Number of entries. The number of entries that are contained within the specific list of information returned for a given key.

» **Number of entries behind.** If the remote journal delivery mode is *ASYNC, this is the number of entries that are waiting to be sent to the target system.

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field is -1.

Number of internet addresses. The number of internet addresses configured for data port services.

This field will be -1 when the remote journal is not configured for data port services. If this field is -1, the relational database is used for all communications to the target system. «

Number of keys in key section. The number of keys that are listed in the key section with specific information returned.

Object file identifier. The file identifier (FID) of the integrated file system object that is journaled to the specified journal.

This field will be blank if the object type field is not the special value *IFS.

Object library name. The name of the library that contains the object that is journaled to the specified journal.

This field will be contain *DIR, *SYMLNK, or *STMF if the object type field is the special value *IFS.

Object name. The name of the object that is journaled to the specified journal.

This field will be blank if the object type field is the special value *IFS.

Object type. The type of the object that is journaled to the specified journal.

Note: If the object is of type *DIR, *SYMLNK, or *STMF, the object type listed will be the special value *IFS.

» **Offset to internet address array.** The byte offset from the start of the **Relational database dirctory entry** field to the start of the **Internet address** array.

This field will be -1 when the remote journal not configured for data port services. If this field is -1, the relational database is used for all communications to the target system. «

Offset to key information. The offset from the start of the format to the key information section. Specifically, this offset points to the Number of keys in key section field.

Offset to start of key information. The offset from the start of the key section, which starts immediately after the Number of keys in key section field, to the specific information returned for a given key.

Receiver size option *MAXOPT1. Whether the journal receiver attached to the journal can have a maximum receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 9,999,999,999. Additionally, the maximum size of the journal entry that can be deposited is 15,761,440 bytes. Journal receivers attached to a journal while this option is in effect cannot be saved and restored to any releases prior to V4R5M0, nor can they be replicated to any remote journals on any systems at releases prior to V4R5M0. This option is applicable only for local journals and is blank for remote journals.

blank Blank is returned when the journal is a remote journal.

0 The journal receivers attached to the journal will have a maximum journal receiver size of approximately 1.9 gigabytes and a maximum sequence number of 2,147,483,136, if neither *MAXOPT2 nor *MAXOPT3 is specified.

1 The journal receivers attached to the journal will have a maximum journal receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 9,999,999,999. Additionally, the maximum size of the journal entry that can be deposited is 15,761,440 bytes.

Receiver size option *MAXOPT2. Whether the journal receiver attached to the journal can have a maximum receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 9,999,999,999. Additionally, the maximum size of the journal entry which can be deposited is 4,000,000,000 bytes. Journal receivers attached to a journal while this option is in effect cannot be saved and restored to any releases prior to V5R1M0, nor can they be replicated to any remote journals on any systems at releases prior to V5R1M0. This option is applicable only for local journals and is blank for remote journals.

blank Blank is returned when the journal is a remote journal.

0 The journal receivers attached to the journal will have a maximum journal receiver size of approximately 1.9 gigabytes and a maximum sequence number of 2,147,483,136, if neither *MAXOPT1 nor *MAXOPT3 is specified.

1 The journal receivers attached to the journal will have a maximum journal receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 9,999,999,999. Additionally, the maximum size of the journal entry that can be deposited is 4,000,000,000 bytes.

Receiver size option *MAXOPT3. Whether the journal receiver attached to the journal can have a maximum receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 18,446,744,073,709,551,600. Additionally, the maximum size of the journal entry which can be deposited is 4,000,000,000 bytes. Journal receivers attached to a journal while this option is in effect cannot be saved and restored to any releases prior to V5R3M0, nor can they be replicated to any remote journals on any systems at releases prior to V5R3M0. This option is applicable only for local journals and is blank for remote journals.

blank Blank is returned when the journal is a remote journal.

0 The journal receivers attached to the journal will have a maximum journal receiver size of approximately 1.9 gigabytes and a maximum sequence number of 2,147,483,136, if neither *MAXOPT1 nor *MAXOPT2 is specified.

1 The journal receivers attached to the journal will have a maximum journal receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 18,446,744,073,709,551,600. Additionally, the maximum size of the journal entry that can be deposited is 4,000,000,000 bytes.

Receiver size option *MINFIXLEN. The size of the journal entries that are deposited into the attached journal receivers is reduced by the automatic removal of all fixed length data such as job name, machine sequence number, and so on. This option is applicable only for local journals and is blank for remote journals.

blank Blank is returned when the journal is a remote journal.

0 The journal entries that are deposited include all of the fixed length data such as job name, system sequence number, and so on.

1 The journal entries that are deposited do not include any of the fixed length data such as job name, system sequence number, and so on.

Receiver size option *RMVINTENT. Whether the size of the receivers that are attached to the journal are reduced by automatic removal of the internal system entries. Removal occurs only for entries that are required for initial program load (IPL) recovery when those entries are no longer required. This option is applicable only for local journals and is blank for remote journals.

blank Blank is returned when the journal is a remote journal.

0 The internal system entries are not automatically removed when they are no longer needed for recovery.

1 The internal system entries are automatically removed when they are no longer needed for recovery.

Redirected receiver library name. For a local or *TYPE1 remote journal, this field gives the redirected receiver library name that is currently in place on this journal's local journal for any downstream journal receivers associated with *TYPE1 remote journals.

This field is set to *NONE if no receiver library redirection was specified when *TYPE1 remote journals were added.

This field is set to the redirected receiver library name that is currently in place on this remote journal if the specified journal is a *TYPE2 remote journal.

Relational database directory entry. The name of the relational database directory entry that is associated with the remote journal.

Relational database directory entry details. The first 512 bytes of the relational database (RDB) entry details being used to communicate to the target system. To view the format of this information use the Display File Field Description (DSPFFD) command on the RDB directory logical file, QADBXRMTNM, in library QSYS.

To see details beyond what is returned in the first 512 bytes, specify the RDB name on the Display Relational Database Entry Detail (DSPRDBDIRE) command.

Remote journal active state time. The date and time that the remote journal environment transitioned from a *SYNCPEND or *ASYNCPEND delivery mode to a *SYNC or *ASYNC delivery mode. This field is in the CYYMMDDHHMMSS format as follows:

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

If the remote journal state is *INACTIVE, this field will be blank.

Remote journal last catch-up time. The date and time that the remote journal environment last transitioned to a *SYNCPEND or *ASYNCPEND delivery mode. This field is in the CYYMMDDHHMMSS format as follows:

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

If the remote journal state is *INACTIVE, this field will be blank.

Remote journal delivery mode. The remote journal delivery mode that is being used to replicate journal entries to the remote journal.

0	Not applicable. The remote journal is not *ACTIVE or not *CTLINACT.
1	*ASYNC. Journal entries are being delivered or replicated asynchronously.
2	*SYNC. Journal entries are being delivered or replicated synchronously.
3	*ASYNCPEND. Journal entries are to be delivered or replicated asynchronously, but the remote journal is currently in catch-up mode.
4	*SYNCPEND. Journal entries are to be delivered or replicated synchronously, but the remote journal is currently in catch-up mode.

Remote journal library name. The library name of the remote journal that is directly downstream of this journal.

This field is blank if there is no remote journal.

Remote journal name. The name of the remote journal that is directly downstream of this journal.

This field is blank if there is no remote journal.

Remote journal receiver library name. The library name of the remote journal receiver that is directly downstream of this journal.

This field is blank if there is no remote journal or if one was not specified on the Add Remote Journal (ADDRMTJRN) command or the Add Remote Journal (QjoAddRemoteJournal) API.

Remote journal state. An indication as to whether the remote journal is actively receiving journal entries from the source system journal.

0 *INACTIVE. The remote journal is not ready to receive any journal entries from its source journal.

1 *ACTIVE. The remote journal is ready to receive any journal entries from its source journal.

2 *FAILED. The remote journal is not ready to receive any journal entries from its source journal due to a remote journal function failure, for example, a communications failure. You will need to inactivate the remote journal by using the Change Remote Journal (CHGRMTJRN) command or by calling the Change Journal State (QjoChangeJournalState) API.

3 *CTLINACT. The remote journal is in the process of a controlled inactivate. Therefore, the remote journal will be receiving those journal entries that were already queued for replication when the Change Remote Journal (CHGRMTJRN) command or the Change Journal State (QjoChangeJournalState) API requested to inactivate the remote journal. However, no entries deposited after that request will be replicated to the remote journal.

4 *PENDING. The remote journal is transitioning from an *INACTIVE state to an *ACTIVE state. If the system abnormally ends while in a *PENDING state, the *PENDING state will be displayed until the next attempt to activate the remote journal.

Remote journal type. The type of remote journal that was created, and that influences characteristics of the remote journal such as journal receiver restore options, redirection capabilities, and remote journal association support. The possible values are:

0 Local journal

1 *TYPE1 remote journal

2 *TYPE2 remote journal

Reserved. The bytes reserved to align binary fields or for future use.

Sending task priority. If the remote journal delivery mode is *ASYNC, this is the priority of the sending task on the source system.

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field is -1.

Source journal ASP group name. The name of the independent auxiliary storage pool (ASP) group of the source journal. *SYSBAS is used to indicate the system ASP and all basic user ASPs. The source journal is the journal that is directly upstream from this journal.

This field is blank if there is no source journal.

Source journal library name. The library name of the source journal. The source journal is the journal that is directly upstream of this journal.

This field is blank if there is no source journal, if no journal receiver is currently attached, or if the journal receiver is damaged or destroyed.

Source journal name. The journal name of the source journal. The source journal is the journal that is directly upstream of this journal.

This field is blank if there is no source journal.

This field is *UNKNOWN if no journal receiver is currently attached, or if the journal receiver is damaged.

Source journal system. The system name of the source journal. The source journal is the journal that is directly upstream of this journal.

This system name is determined when the remote journal is activated by using the Change Remote Journal (CHGRMTJRN) command or by calling the Change Journal State (QjoChangeJournalState) API. The name is based on the current system name at that time as seen by using the Display Network Attributes (DSPNETA) command.

This field is blank if there is no source journal, if no journal receiver is currently attached, or if the journal receiver is damaged.

Source journal system associated with the attached journal receiver. If this attached journal receiver is associated with a remote journal, this field is the system name of the source journal. See the source journal system field for more information.

Source journal system associated with the journal receiver. If this journal receiver was associated with a remote journal, this field is the system name of the source journal. See the source journal system field for more information.

» **Super bundle count.** If the remote journal delivery mode is *ASYNC, this is the number of that the remote journal environment has automatically gone into super bundling mode. Super bundling mode helps the remote journal environment keep up with the local journal when the local journal has a high rate of journal entry deposits.

If the remote journal delivery mode is not *ASYNC, or the remote journal state is *INACTIVE, this field is -1.

Synchronous sending time out. If the remote journal delivery mode is *SYNC, this is the maximum amount of time in seconds to wait for a response from the remote system when a response is required in a synchronous remote journal environment. A special value of 0 indicates that the system will choose a system default for the synchronous sending time out.

If the remote journal delivery mode is not *SYNC, or the remote journal state is *INACTIVE, this field is -1. «

Total number of journaled access paths. The total number of access paths that are currently being journaled to this journal.

Total number of journaled commitment definitions. The total number of commitment definitions that are currently being implicitly journaled to this journal.

Total number of journaled data areas. The total number of data areas that are currently being journaled to this journal.

Note: This field will be set in the key 2 return structure only if data areas were requested to be returned by the journaled object information key. Otherwise it will be 0 in the key 2 return structure. This value will always be returned in the base format structure.

Total number of journaled data queues. The total number of data queues that are currently being journaled to this journal.

Note: This field will be set in the key 2 return structure only if data queues were requested to be returned by the journaled object information key. Otherwise it will be 0 in the key 2 return structure. This value will always be returned in the base format structure.

Total number of journaled files. The total number of files that are currently being journaled to this journal.

Note: This field will be set in the key 2 return structure only if files were requested to be returned by the journaled object information key. Otherwise it will be 0 in the key 2 return structure. This value will always be returned in the base format structure.

Total number of journaled integrated file system objects of type *DIR, *STMF, and *SYMLNK. The total number of integrated file system objects of type *DIR, *STMF, and *SYMLNK that are currently being journaled to this journal.

Note: This field will be set in the key 2 return structure only if integrated file system objects were requested to be returned by the journaled object information key. Otherwise it will be 0 in the key 2 return structure. This value will always be returned in the base format structure.

» **Total number of journaled libraries.** The total number of libraries that are currently being journaled to this journal.

Note: This field will be set in the key 2 return structure only if libraries were requested to be returned by the journaled object information key. Otherwise it will be 0 in the key 2 return structure. This value will always be returned in the base format structure. «

Total number of journaled members. The total number of file members that are currently being journaled to this journal.

Note: This field will be set in the key 2 return structure only if files were requested to be returned by the journaled object information key. Otherwise it will be 0 in the key 2 return structure. This value will always be returned in the base format structure.

Total number of journaled objects. This is the total of all objects journaled to the journal. This count includes explicitly journaled objects such as files, file members, access paths, data areas, data queues, » libraries «, and integrated file system objects. This count also includes implicitly journaled objects such as journal receivers, commitment definitions, and objects journaled for system recovery purposes.

Note: The summation of these total number of object fields may give a number that is higher or lower than the value returned in the **total number of journaled objects** field. This is possible because the total number of objects includes some objects journaled by the system for system recovery purposes. Also, all of the total fields can be actively changing while retrieving the values.

Total number of journal receivers. The total number of journal receivers that are associated with the journal.

Total number of remote journals. The total number of remote journals that are directly downstream of this journal.

Total size of journal receivers. The total size of the journal receivers in number of kilobytes of auxiliary disk storage that are associated with the journal. The size is in units of the total size of journal receivers multiplier. The total size is equal to or smaller than the total size multiplied by the total size of journal receivers multiplier.

Total size of journal receivers multiplier. The value to multiply the total size of journal receivers by to get the true total size. The value is 1 if the total size of journal receivers is smaller than 2,147,483,647 kilobytes and 1024 if it is larger.

» **Validity checking.** The validity checking status. When communications validity checking is turned on, the remote journal environment will provide additional checking to verify that the data which is received by the target system matches the data that was sent from the source system. If the data does not match, the data will not be written to the target system, the remote journal environment will be inactivated, and messages indicating the communications failure will be issued to the journal message queue and QHST. The possible values follow:

blank Blank is returned when the journal is not sending entries to a target system.
0 Communications validity checking is turned off for this remote journal environment.
1 Communications validity checking is turned on for this remote journal environment.

Note: Communications validity checking may impact performance.

«

Error Messages

Message ID	Error Message Text
» CPFB8ED E	Device description &1 not correct for operation. «
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.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF694B E	Length &1 of variable record for key &2 not valid.
CPF694C E	Variable length record data for key &1 not valid.
CPF6948 E	Length of the receiver variable &1 is not valid.
» CPF69A8 E	Device description &1 not valid for operation. «
CPF69A9 E	Internal error detected, error code &2.
CPF701B E	Journal recovery of interrupted operation failed.
CPF702C E	An attached receiver has previously been destroyed.
CPF708D E	Journal receiver found logically damaged.
CPF8100 E	All CPF81xx messages could be returned. 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.
CPF9810 E	Library &1 not found.
» CPF9814 E	Device &1 not found. «
CPF9820 E	Not authorized to use library &1.
» CPF9825 E	Not authorized to device &1. «
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Qualified journal receiver name	Input	Char(20)
4	Format name	Input	Char(8)

Omissible Parameter:

5	Error code	I/O	Char(*)

Service Program Name: QJOURNAL
Header File: QSYSINC/H.QJOURNAL
Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API provides access to journal-receiver-related information to help manage a journal environment, including a remote journal environment.

Various types of journal receiver information are provided by the API, similar to information reported using the Display Journal Receiver Attributes (DSPJRNRCVA) CL command.

Authorities and Locks

Journal Receiver Authority
*OBJOPR and some data authority other than *EXECUTE

Journal Receiver Library Authority
*EXECUTE

Journal Authority
*OBJOPR

Journal Library Authority
*EXECUTE

Service Program Authority
*EXECUTE

Journal Receiver Lock
*SHRRD

Journal Lock
*SHRRD

For the following authorities and locks:
- Journal Authority
- Journal Library Authority
- Journal Lock

These are required only if the journal receiver has ever been associated with a journal, and that journal is on the system.

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

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

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable 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.

Qualified journal receiver name
INPUT; CHAR(20)

The name of the journal receiver and its library from which the journal receiver attributes and information is to be retrieved. The first 10 characters contain the journal receiver name, and the second 10 characters contain the library name. The special values supported for the library name follow:

**LIBL*	Library list
**CURLIB*	Current library

Format name
INPUT; CHAR(8)

The format RRCV0100 is the only supported format that is used by this API. For more information, see "RRCV0100 Format."

Omissible 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.

RRCV0100 Format

The structure of the information returned is determined by the specified format name. For detailed descriptions of the fields, see "Field Descriptions" on page 134.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Journal receiver name
18	12	CHAR(10)	Journal receiver library name
28	1C	CHAR(10)	Journal name
38	26	CHAR(10)	Journal library name
48	30	BINARY(4)	Threshold

Offset		Type	Field
Dec	Hex		
52	34	BINARY(4)	Size
56	38	BINARY(4)	Auxiliary storage pool (ASP)
60	3C	BINARY(4)	Number of journal entries
64	40	BINARY(4)	Maximum entry-specific data length
68	44	BINARY(4)	Maximum null value indicators
72	48	BINARY(4)	First sequence number
76	4C	CHAR(1)	Minimize entry specific data for data areas
77	4D	CHAR(1)	Minimize entry specific data for files
78	4E	CHAR(2)	Reserved
80	50	BINARY(4)	Last sequence number
84	54	BINARY(4)	Reserved
88	58	CHAR(1)	Status
89	59	CHAR(1)	Receiver size option *MINFIXLEN
90	5A	CHAR(1)	Receiver maximums option
91	5B	CHAR(4)	Reserved
95	5F	CHAR(13)	Attached date and time
108	6C	CHAR(13)	Detached date and time
121	79	CHAR(13)	Saved date and time
134	86	CHAR(50)	Text
184	B8	CHAR(1)	Pending transactions
185	B9	CHAR(1)	Remote journal type
186	BA	CHAR(10)	Local journal name
196	C4	CHAR(10)	Local journal library name
206	CE	CHAR(8)	Local journal system
214	D6	CHAR(10)	Local journal receiver library name
224	E0	CHAR(10)	Source journal name
234	EA	CHAR(10)	Source journal library name
244	F4	CHAR(8)	Source journal system
252	FC	CHAR(10)	Source journal receiver library name
262	106	CHAR(10)	Redirected journal receiver library
272	110	CHAR(10)	Dual journal receiver name
282	11A	CHAR(10)	Dual journal receiver library name
292	124	CHAR(10)	Previous journal receiver name
302	12E	CHAR(10)	Previous journal receiver library name
312	138	CHAR(10)	Previous dual journal receiver name
322	142	CHAR(10)	Previous dual journal receiver library name
332	14C	CHAR(10)	Next journal receiver name
342	156	CHAR(10)	Next journal receiver library name
352	160	CHAR(10)	Next dual journal receiver name
362	16A	CHAR(10)	Next dual journal receiver library name

Offset		Type	Field
Dec	Hex		
372	174	CHAR(20)	Number of journal entries - long
392	188	CHAR(20)	Maximum entry-specific data length - long
412	19C	CHAR(20)	First sequence number - long
432	1B8	CHAR(20)	Last sequence number - long
452	1C4	CHAR(10)	ASP device name
462	1CE	CHAR(10)	Local journal ASP group name
472	1D8	CHAR(10)	Source journal ASP group name
482	1E2	CHAR(1)	Fixed length data *JOB
483	1E3	CHAR(1)	Fixed length data *USR
484	1E4	CHAR(1)	Fixed length data *PGM
485	1E5	CHAR(1)	Fixed length data *PGMLIB
486	1E6	CHAR(1)	Fixed length data *SYSSEQ
487	1E7	CHAR(1)	Fixed length data *RMTADR
488	1E8	CHAR(1)	Fixed length data *THD
489	1E9	CHAR(1)	Fixed length data *LUW
490	1EA	CHAR(1)	Fixed length data *XID
491	1EB	CHAR(21)	Reserved

Field Descriptions

Attached date and time. The date and time that this journal receiver was attached to the journal. For a journal receiver attached to a *REMOTE journal, this is the date and time that the journal receiver was attached on the local system. This field is in the CYYMMDDHHMMSS format as follows:

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

Auxiliary storage pool (ASP). The number of the auxiliary storage pool to which storage for the object is allocated.

ASP device name. The name of the independent auxiliary storage pool (ASP) device to which storage for the object is allocated. *SYSBAS is used to indicate the system ASP and all basic user ASPs.

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.

Detached date and time. The date and time that this journal receiver was detached from the journal. For a journal receiver attached to a *REMOTE journal, this is the date and time that the journal receiver was detached on the local system. This field is in the CYYMMDDHHMMSS format, which is described in the attached date and time field description.

Dual journal receiver library name. The name of the library that contains the dual journal receiver.

This field is blank if there is no dual receiver.

Dual journal receiver name. The journal receiver that was attached at the same time as the journal receiver.

This field is blank if there is no dual receiver.

First sequence number. The journal sequence number of the first journal entry in this journal receiver.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will be in the First sequence number - long field.

First sequence number - long. This is the same field as First sequence number except the information is in a Char(20) field which is treated as Zoned(20,0).

Fixed length data *JOB. Indicates whether the job name will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the job name.
1	Journal entries deposited to the journal will include the job name.

Fixed length data *LUW. Indicates whether the logical unit of work identifier will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal
0	Journal entries deposited to the journal will not include the logical unit of work identifier.
1	Journal entries deposited to the journal may include the logical unit of work identifier.

Fixed length data *PGM. Indicates whether the program name will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the program name.
1	Journal entries deposited to the journal will include the program name.

Fixed length data *PGMLIB. Indicates whether the program library name will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the program library name and the library ASP information.
1	Journal entries deposited to the journal will include the program library name and the library ASP information.

Fixed length data *RMTADR. Indicates whether the remote address will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the remote address.
1	Journal entries deposited to the journal may include the remote address.

Fixed length data *SYSSEQ. Indicates whether the system sequence number will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the system sequence number.
1	Journal entries deposited to the journal will include the system sequence number.

Fixed length data *THD. Indicates whether the thread identifier will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the thread identifier.
1	Journal entries deposited to the journal will include the thread identifier.

Fixed length data *USR. Indicates whether the user name will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the user name.
1	Journal entries deposited to the journal will include the user name.

Fixed length data *XID. Indicates whether the transaction identifier will be stored when journal entries are deposited.

blank	Blank is returned when the journal is a remote journal.
0	Journal entries deposited to the journal will not include the transaction identifier.
1	Journal entries deposited to the journal may include the transaction identifier.

Journal library name. The name of the library in which the journal is stored.

This field is blank if this journal receiver is not yet associated with any journal.

Journal name. The name of the journal.

This field is *NONE if this journal receiver is not yet associated with any journal.

Journal receiver library name. The name of the library that contains the journal receiver.

Journal receiver name. The name of the journal receiver.

Last sequence number. The journal sequence number of the last journal entry in this journal receiver.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will be in the Last sequence number - long field.

Last sequence number - long. This is the same field as Last sequence number except the information is in a Char(20) field which is treated as Zoned(20,0).

Local journal ASP group name. The name of the independent auxiliary storage pool (ASP) group of the local journal. *SYSBAS is used to indicate the system ASP and all basic user ASPs. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

Local journal library name. The library name of the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

Local journal name. The journal name of the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

This field is *NONE if this journal receiver is not yet associated with any journal.

Local journal receiver library name. The library name of the local journal receiver that is associated with the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This field is blank if there is no local journal.

This field is *NONE if this journal receiver is not yet associated with any journal.

Local journal system. The system name of the local journal. The local journal is the journal that is the initiator of the original journal deposit that has been replicated downstream to this journal.

This system name is determined when the remote journal is activated by using the Change Remote Journal (CHGRMTJRN) command or by calling the Change Journal State (QjoChangeJournalState) API. The name is based on the current system name at that time as seen by using the Display Network Attributes (DSPNETA) command.

This field is blank if there is no local journal.

Maximum entry-specific data length. The length in bytes of the longest entry-specific data among all journal entries in this journal receiver.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will be in the Maximum entry-specific data length - long field.

Maximum entry-specific data length - long. This is the same field as Maximum entry-specific data length except the information is in a Char(20) field which is treated as Zoned(20,0).

Maximum null value indicators. The maximum number of null value indicators among all journal entries in this journal receiver.

Minimize entry specific data for data areas. Whether the receiver was attached when MINENTDTA(*DTAARA) was in effect for the journal. Possible values follow:

0 Journal entries for data areas will have complete entry specific data.
1 Journal entries for data areas may have minimized entry specific data.

If this journal receiver is associated with a remote journal, then the value for this field was determined by the local journal.

Minimize entry specific data for files. Whether the receiver was attached when MINENTDTA(*FILE) was in effect for the journal. Possible values follow:

0 Journal entries for files will have complete entry specific data.
1 Journal entries for files may have minimized entry specific data. The minimizing does not occur on field boundaries. Therefore, the entry specific data may not be viewable and may not be used for auditing purposes.
2 Journal entries for files may have minimized entry specific data. The minimizing occurs on field boundaries. Therefore, the entry specific data will be viewable and may be used for auditing purposes.

If this journal receiver is associated with a remote journal, then the value for this field was determined by the local journal.

Next dual journal receiver library name. The library name of the next dual journal receiver that is associated with this journal receiver.

This field is blank if there is no next dual journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Next dual journal receiver name. The name of the next dual journal receiver that is associated with this journal receiver.

This field is blank if there is no next dual journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Next journal receiver library name. The library name of the next journal receiver that is associated with this journal receiver.

This field is blank if there is no next journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Next journal receiver name. The name of the next journal receiver that is associated with this journal receiver.

This field is blank if there is no next journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Number of journal entries. The number of journal entries that are contained in this journal receiver.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will be in the Number of journal entries - long field.

Number of journal entries - long. This is the same field as Number of journal entries except the information is in a Char(20) field which is treated as Zoned(20,0).

Pending transactions. Whether the journal receiver contains journal entries for commitment control transactions that have not yet been committed or rolled back. The possible values are:

0 The journal receiver does not contain entries for pending commitment control transactions.
1 The journal receiver contains entries for pending commitment control transactions.

Previous dual journal receiver library name. The library name of the previous dual journal receiver that is associated with this journal receiver.

This field is blank if there is no previous dual journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Previous dual journal receiver name. The name of the previous dual journal receiver that is associated with this journal receiver.

This field is blank if there is no previous dual journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Previous journal receiver library name. The library name of the previous journal receiver that is associated with this journal receiver.

This field is blank if there is no previous journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Previous journal receiver name. The name of the previous journal receiver that is associated with this journal receiver.

This field is blank if there is no previous journal receiver, or if the specified journal receiver is currently associated with a remote journal.

Receiver maximums option. Indicates the journal receiver sequence number and size option for this journal receiver. Possible values follow:

blank	The journal receiver has not yet been attached to any journal.
0	The journal receiver has a maximum journal receiver size of approximately 1.9 gigabytes and a maximum sequence number of 2,147,483,136.
1	The journal receiver has a maximum journal receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 9,999,999,999. Additionally, the maximum size of the journal entry that can be deposited is 15,761,440 bytes. This occurs if this receiver was attached when RCVSIZOPT(*MAXOPT1) was in effect for the journal. These journal receivers cannot be saved and restored to any releases prior to V4R5M0, nor can they be replicated to any remote journals on any systems at releases prior to V4R5M0.
2	The journal receiver has a maximum journal receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 9,999,999,999. This occurs if this receiver was attached when RCVSIZOPT(*MAXOPT2) was in effect for the journal. Additionally, the maximum size of the journal entry which can be deposited is 4,000,000,000 bytes. These journal receivers cannot be saved and restored to any releases prior to V5R1M0, nor can they be replicated to any remote journals on any systems at releases prior to V5R1M0.
3	The journal receiver has a maximum journal receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 18,446,744,073,709,551,600. This occurs if this receiver was attached when RCVSIZOPT(*MAXOPT3) was in effect for the journal. Additionally, the maximum size of the journal entry which can be deposited is 4,000,000,000 bytes. These journal receivers cannot be saved and restored to any releases prior to V5R3M0, nor can they be replicated to any remote journals on any systems at releases prior to V5R3M0.

If this journal receiver is associated with a remote journal, then the value for this field was determined by the local journal.

Receiver size option *MINFIXLEN. The size of the journal entries that are deposited into the attached journal receivers is reduced by the automatic removal of all fixed length data such as job name, system sequence number, and so on. This option is applicable only for local journals and is blank for remote journals.

blank	Blank is returned when the journal is a remote journal.
0	The journal entries in the receiver include all of the fixed length data such as job name, system sequence number, and so on.
1	The journal entries in the receiver do not include any fixed length data such as job name, system sequence number, and so on.

Redirected journal receiver library. If this journal receiver was attached to a remote journal, this field is the *TYPE1 receiver library redirection that was in effect when this journal receiver was attached.

This field is blank if this journal receiver was attached to a local journal, or if the journal receiver was attached to a remote journal with no *TYPE1 receiver library redirection.

This field is *NONE if this journal receiver is not yet associated with any journal.

Remote journal type. If this journal receiver was attached to a remote journal, this field is the remote journal type for that journal, when this journal receiver was attached. The possible values are:

blank	The journal receiver has not yet been attached to any journal.
0	The journal receiver was attached to a local journal.
1	The journal receiver was attached to a *TYPE1 remote journal.
2	The journal receiver was attached to a *TYPE2 remote journal.

Reserved. The bytes reserved to align binary fields or for future use.

Saved date and time. The date and time that the journal receiver was last saved. This value reflects when the receiver was saved from the system it exists on (either the source or target system time). This field is in the CYYMMDDHHMMSS format, which is described in the attached date and time field description.

Size. The number of kilobytes of auxiliary disk storage used by this journal receiver.

Source journal ASP group name. The name of the independent auxiliary storage pool (ASP) group of the source journal. *SYSBAS is used to indicate the system ASP and all basic user ASPs. The source journal is the journal that is directly upstream of this journal.

This field is blank if there is no source journal.

Source journal library name. The library name of the source journal. The source journal is the journal that is directly upstream of this remote journal.

This field is blank if there is no source journal.

Source journal name. The journal name of the source journal. The source journal is the journal that is directly upstream of this remote journal.

This field is blank if there is no source journal.

This field is *NONE if this journal receiver is not yet associated with any journal.

Source journal receiver library name. The library name of the source journal receiver that is associated with the source journal. The source journal is the journal that is directly upstream of this remote journal.

This field is blank if there is no source journal.

This field is *NONE if this journal receiver is not yet associated with any journal.

Source journal system. The system name of the source journal. The source journal is the journal that is directly upstream of this remote journal.

This system name is determined when the remote journal is activated by using the Change Remote Journal (CHGRMTJRN) command or by calling the Change Journal State (QjoChangeJournalState) API, and is based on the current system name at that time as seen by using the Display Network Attributes (DSPNETA) command.

This field is blank if there is no source journal.

Status. The status of the journal receiver. The status can be one of the following:

1 The journal receiver is currently attached to the journal.

2 The journal receiver is online. The journal receiver has not been saved, and it has been detached from the journal.

3 The journal receiver was saved after it was detached. The journal receiver storage was not freed when it was saved.

4 The journal receiver was saved after it was detached. The journal receiver storage was freed when it was saved.

5 The journal receiver status is partial for one of the following reasons:

- It was restored from a version that was saved while it was attached to the journal. Additional journal entries may have been written that were not restored.
- It was one of a pair of dual journal receivers, and it was found damaged while attached to the journal. The journal receiver has since been detached. This journal receiver is considered partial because additional journal entries may have been written to the dual journal receiver.
- It is associated with a remote journal and it does not contain all the journal entries that are in the associated journal receiver attached to the source journal.

6 The journal receiver status is empty, since the receiver has never been attached to a journal.

Text. The text description of the journal receiver.

Threshold. An auxiliary disk storage space threshold value (in kilobytes) for the journal receiver. If the threshold value is exceeded during journaling and the journal has the MNGRCV(*USER) attribute, a message (CPF7099) is sent to the message queue that is specified on the Create Journal (CRTJRN) or the Change Journal (CHGJRN) command. If the journal has the MNGRCV(*SYSTEM) attribute, the system creates and attaches a new journal receiver, detaches the old journal receiver when the threshold is reached, and sends message CPF7020 to the journal message queue. This option is applicable only for local journals and is zero for remote journals.

Error Messages

Message ID	Error Message Text
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.
CPF3C90 E	Literal value cannot be changed.
CPF6948 E	Length of the receiver variable &1 is not valid.
CPF701A E	Journal receiver not eligible for operation.
CPF701B E	Journal recovery of interrupted operation failed.
CPF8100 E	All CPF81xx messages could be returned. 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.
CPF9804 E	Object &2 in library &3 damaged.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Return LU6.2 Partners (QTNRTNLU) API

Optional Parameter:

1	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Return LU6.2 Partners (QTNRTNLU) API identifies all the LU6.2 partner logical unit (LU) names that are known to this system. The following informational message is sent to the joblog of the job issuing the API to identify each partner:

CPI83D1 Partner LU &1.&2 is known to this system.

Replacement variable &1 is the partner LU remote network identifier and &2 is the partner LU location name.

Connection problems sometimes occur after a partner LU is moved to a backup system with the same SNA configuration as the original system. The "Clear LU6.2 Partners (QTNCLRLU) API" on page 32 can be used to correct such problems.

Authorities and Locks

Authority
*ALLOBJ special authority is required.

Locks None.

Required Parameter

None

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 caller of the API.

Usage Notes

This API was designed so that it would be easy to use from a CL command line. The following CL command will return all the LU6.2 partners known to this system:

```
CALL PGM(QTNRTNLU)
```

Error Messages

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF83ED E	&1 API requires &2 special authority.

Message ID	Error Message Text
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R3

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Rollback Required (QTNRBRQD) API

Required Parameter Group:

1	Resource handle	Input	Binary(4)
2	Error code	I/O	Char(*)

Threadsafe: Yes

The Rollback Required (QTNRBRQD) API puts the commitment definition currently active for the activation group of the program making the request into a rollback-required state. When a commitment definition is in a rollback-required state, protected resources cannot be used until a rollback operation is performed.

Authorities and Locks

None.

Required Parameter Group

Resource handle

INPUT; BINARY(4)

The resource handle returned by the Add Commitment Resource (QTNADDCR) API when the API commitment resource was added to the current commitment definition.

Error code

I/O; CHAR(*)

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

Restrictions

You are prevented from putting the current commitment definition in a rollback-required state when:

- The resource handle is not valid.
- Commitment control is not active for the program making the request to put the commitment definition into a rollback required state.
- Commit or rollback processing is in progress for the current commitment definition.

In all other instances, the current commitment definition is put in a rollback-required state.

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.
CPF8362 E	Request for commit resource is not valid; reason code &1.
CPF8367 E	Cannot perform commitment control operation.

API introduced: V2R3

Send Journal Entry (QJOSJRNE) API

Required Parameter Group:

1	Qualified journal name	Input	Char(20)
2	Journal entry information	Input	Char(*)
3	Entry data	Input	Char(*)
4	Length of entry data	Input	Binary(4)
5	Error Code	I/O	Char(*)

Optional Parameter Group 1:

6	Receiver variable	Output	Char(*)
7	Length of receiver variable	Input	Binary(4)
8	Format of receiver variable	Input	Char(8)
9	Minimum length of entry data returned	Input	Binary(4)

Default Public Authority: *USE
Threadsafe: Yes

The Send Journal Entry (QJOSJRNE) API writes a single journal entry to a specific journal. The format of the entry is determined by the API caller. You can assign an entry type to the journal entry. You can associate the journal entry with additional information such as a journaled object or a commit cycle identifier.

If the journal currently has a state of *STANDBY, then the journal entry will not be deposited unless 1 is specified for the override standby key.

Note: The journal code for the entry is 'U', indicating a user-specified journal entry. See the Journal management topic for more information.

Restrictions

- If an object other than a file is specified, it currently must be journaled to the specified journal.

  If a file object is specified, it currently must either be journaled to the specified journal or it must have been last journaled to the specified journal.
- The specified journal cannot be a remote journal.
- Only one of the following keys can be specified in one call of this API:
  - Keys 2 and 3, qualified file name and member name
  - Key 6, qualified object name
  - Key 7, object path name
  - Key 8, object file identifier

Authorities and Locks

Journal Authority
*OBJOPR and *ADD

Journal Library Authority
*EXECUTE

Non-IFS Object Authority
*OBJOPR

Non-IFS Object Library Authority
*EXECUTE

IFS Object Authority (if present)
*R

IFS Object Directory Authority
*X

File Lock
*SHRNUP

Non-IFS Object, other than File, Lock
*SHRRD

IFS Object Lock (if present)
O_RDONLY | O_SHARE_RDWR

Journal Lock
*SHRUPD

*EXECUTE, *OBJOPR, *R, *X, *SHRUPD, *SHRRD, and O_RDONLY | O_SHARE_RDWR are required only if an object is specified in the qualified file name, qualified object name, object path, or object file identifier key fields.

Required Parameter Group

Qualified journal name
INPUT; CHAR(20)

The name of the journal to which the entry is to be added and the library in which it is located. The first 10 characters contain the journal name and the second 10 characters contain the library name. The special values supported for the library name are:

**LIBL*	Library list
**CURLIB*	Current library

Journal entry information
INPUT; CHAR(*)

Information pertinent to the journal entry that is to be added. 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 information that should be included in the journal entry. For the specific format of the variable length record, see "Format for Variable Length Record" on page 147.

Entry data
INPUT; CHAR(*)

The user-specified data that is placed in the variable portion of the journal entry (also known as entry specific data).

Length of entry data
INPUT; BINARY(4)

The length of the entry data parameter. Valid values are 0 - 15761440. If 0 is specified, this is equivalent to *BLANKS on the SNDJRNE CL command. If the length of the entry data is greater than 32766, then a pointer to the entry data will be returned when retrieving the entry. If the retrieve interface is expecting pointers, the data can be accessed through the pointer returned on the retrieve. Otherwise, the data returned by the retrieve interface will be *POINTER.

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

Receiver variable
OUTPUT; CHAR(*)

The receiver variable that receives the entry information for the journal entry just deposited. You can specify the size of the area smaller than the format requested as long as you specify the length of receiver variable parameter correctly. As a result, the API returns only the data the receiver variable can hold. This parameter is ignored if Length of receiver variable is 0.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable specified in the user program. If the length of the 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 allowed values are 0 or a value of 8 or greater. The default value is 0.

Format of receiver variable
INPUT; CHAR(8)

The following formats are valid inputs: SJNE0000 and SJNE0100. The default value is SJNE0000, which indicates that the Receiver variable parameter is not being used. See the SJNE0100 Format for more information.

Minimum length of entry data returned
INPUT; BINARY(4)

The minimum number of bytes of entry data the caller wants returned on interfaces that retrieve journal information when the length of the entry data is greater than 32766. A pointer will be returned to the remaining information. The default value for this parameter is 0, which indicates that only the pointer will be returned as entry specific data.

If the length of entry data is 32766 or less, the value of the Minimum length of entry data returned parameter must be 0. If the length of entry data is greater than 32766, the user may specify a value of 0 or a multiple of 16 (up to a maximum of 32736) indicating how much of the entry data parameter should be returned as entry specific data prior to a pointer addressing the rest of the entry specific data.

Example: The length of entry data is 40,000 so a pointer to the data will be returned when the journal entry is retrieved. The caller, however, wants the first 16 bytes of the entry data to be returned prior to the pointer and for the pointer to address the remaining 39,984 bytes. This can be done by specifying a Minimum length of entry data returned parameter value of 16. The 16 bytes of user data that are returned as entry specific data could contain information such as the length of data addressed by the pointer.

When the length of entry data is greater than 32766, all of the actual data cannot be returned on interfaces that retrieve journal entry information. Instead, on these interfaces, a pointer is returned to the actual data.

Note: The user of the QJOSJRNE API must pass all or none of the parameters in Optional Parameter Group 1.

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
4	4	BINARY(4)	Length of data
8	8	CHAR(*)	Data

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, an error message (CPF3C4D) 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

Data. The data used to determine how the journal entry should be sent. All values are validity checked.

Key. Identifies specific information about the journal entry that will be sent. See "Keys" for the list of valid keys.

Length of data. The length of the journal entry information value. The length of data field is used to get the addressability of the next attribute record.

Keys

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

Some messages for this API refer to parameters and values of the Send Journal Entry (SNDJRNE) command. This table also can be used to locate the key names that correspond to the SNDJRNE command parameters.

Key	Type	Field	SNDJRNE Command Parameter
1	CHAR(2)	Journal entry type	TYPE
2	CHAR(20)	Qualified file name	FILE
3	CHAR(10)	Member name	FILE
4	CHAR(1)	Force journal entry	FORCE
5	CHAR(1)	Include commit cycle identifier	Not applicable
6	CHAR(40)	Qualified object name	OBJ

Key	Type	Field	SNDJRNE Command Parameter
7	CHAR(*)	Object path name	OBJPATH
8	CHAR(16)	Object file identifier	OBJFID
9	CHAR(1)	Override journal state	OVRSTATE

Field Descriptions

Force journal entry. Whether the journal receiver is forced to auxiliary storage after the user entry is written to it. Possible values are:

0 The journal receiver is not forced to auxiliary storage. This is the default value if the key is not specified.
1 The journal receiver is forced to auxiliary storage.

Include commit cycle identifier. Whether the commit cycle identifier should be included with this user journal entry when it is written. The commit cycle identifier will be the one associated with the commitment definition that is being used by the program that calls this API. Possible values are:

0 The commit cycle identifier is not included and will not be written with this user journal entry. This is the default value if the key is not specified.
1 The commit cycle identifier is determined and will be associated with this user journal entry if it is available. So that the commit cycle identifier can be determined for the specified journal, you *must* have registered an API commitment resource with the Add Commitment Resource (QTNADDCR) API. Also, you must have specified that this journal was associated with this commitment resource. For more information, see "Add Commitment Resource (QTNADDCR) API" on page 3 (QTNADDCR) API.

Note: If commitment control is not active for the program that calls this API, an error will be returned as no commit cycle identifier is available. You can use the "Retrieve Commitment Information (QTNRCMTI) API" on page 58 (QTNRCMTI) API to determine whether commitment control is active or not for the commitment definition of the program that calls this API.

Notes:

1. If QJOSJRNE is called during commitment control IPL recovery, no commit cycle identifier is available to be included. Therefore, during this IPL recovery, the journal entry will be sent without a commit cycle identifier, no matter which value is specified.
2. For more information on commitment definitions and commit cycle identifiers, see the Journal management topic.

Journal entry type. The journal entry type of this journal entry. Specify a 2-character value for the journal entry type. This value must be greater than or equal to hex C000. A default value of '00' (hex F0F0) is assumed if the key is not specified.

If a hexadecimal value is specified that does not represent characters, that value is not shown on the DSPJRN display or printout.

Member name. The name of the physical file member with which this entry is associated. Special values are *FIRST and *NONE. The default value is *FIRST. If file name is *NONE and this field has a specific member listed, an error will be returned.

Object file identifier. The file identifier (FID) of the object with which this entry is associated. An FID is a unique identifier associated with integrated file system-related objects. Only objects of type *STMF, *DIR, or *SYMLNK that are in the "root" (/), QOpenSys, and user-defined file systems are supported.

The only special value supported is 16 bytes of hexadecimal zeros ('00000000000000000000000000000000'X) and represents no object identified by an FID will be associated with the entry. This is the default if the object file identifier key is not specified.

Object path name. The path name of the object with which this entry is associated. Only objects of type *STMF, *DIR, or *SYMLNK that are in the "root" (/), QOpenSys, and user-defined file systems are supported. Symbolic links within the path name will not be followed.

If a pointer is specified in the object path name, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the path name format, see Path name format.

The only special value supported is *NONE, and this is the default if the object path name key is not specified. If *NONE is specified, then in the path name header structure, the length must be set to 5 and *NONE must follow the path name header structure.

The maximum length of data for this key is 16,773,120.

Override journal state. Whether the journal entry will be deposited, overriding the current state of the journal. Possible values are:

0 None of the journal state values are overridden. That is,
- The journal entry is deposited if the journal state is *ACTIVE.
- The journal entry is not deposited and no error is sent if the journal state is *STANDBY.

1 The journal entry is deposited even if the journal state is *STANDBY.

Qualified file name. The first 10 characters contain the name of the physical file with which this entry is associated. The only special value supported for the file name is *NONE. The second 10 characters contain the name of the library containing the physical file. Special values are:

**LIBL*	Library list
**CURLIB*	Current library

If the file name is *NONE, then the library name is ignored. *NONE is the default if the qualified file name key is not specified.

Qualified object name. The qualified name of the object with which this entry is associated. For the format of this field, see "Qualified Object Name Format."

If *NONE is specified for object name, the remaining fields should be set to blanks. *NONE is the default if the qualified file name key is not specified.

Qualified Object Name Format

Offset		Type	Field
Dec	**Hex**		
		CHAR(10)	Object name
		CHAR(10)	Object library name
		CHAR(10)	Object type
		CHAR(10)	Member name, if *FILE specified

Field Descriptions

Member name. The name of the physical file member with which this entry is associated. The possible values are:

**FIRST*	The entry is associated with the first member in the file.
**NONE*	The entry is associated with the file, not with any member of the file.
member name	The name of the file member with which this entry is associated.
blank	The member name field must be blank if *NONE is specified for the object name.

Note: If the specified object type was not *FILE, the member name value is ignored.

Object library name. The name of the library containing the object. 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.
blank	The library name field must be blank if *NONE is specified for the object name.

Object name. The name of the object with which this entry is associated.

**NONE*	No object is associated with the journal entry.
Object name	The name of the object with which this entry is associated.

Object type. The object type associated with the object with which this entry is associated. The possible values are:

**FILE*	The entry is associated with a database file or database file member.
**DTAARA*	The entry is associated with a data area.
**DTAQ*	The entry is associated with a data queue.
**LIB*	The entry is associated with a library.
blank	The object type field must be blank if *NONE is specified for the object name.

SJNE0100 Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(20)	Sequence number
28	1C	CHAR(10)	Journal receiver name
38	26	CHAR(10)	Journal receiver library
48	30	CHAR(10)	Journal receiver ASP device name

Field Descriptions

Bytes available: The length, in bytes, of the SJNE0100 information available for the API to return to the caller.

Bytes returned: The number of bytes that are being returned in the receiver variable.

Journal receiver ASP device name: The name of the ASP device on which the journal receiver resides.

Journal receiver library: The name of the library that contains the journal receiver.

Journal receiver name: The name of the journal receiver that contains the sent entry.

Sequence number: This is the number assigned by the system to the journal entry that was just sent. This is a Char(20) field and it is treated as Zoned(20,0).

Error Messages

Message ID	Error Message Text
CPFA0D4 E	File system error occurred.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
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.
CPF3C85 E	Value for key &1 not allowed with value for key &2.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF6948 E	Length of the receiver variable &1 not valid.
CPF694E E	Value specified for parameter Minimum length of entry data returned not valid.
CPF7002 E	File &1 in library &2 not a physical file.
CPF7003 E	Entry not journaled to journal &1. Reason code &3.
CPF7007 E	Cannot allocate member &3 file &1 in &2.
CPF7037 E	File &1 not journaled to journal &3.
CPF706E E	Length of entry data &1 not valid.
CPF708D E	Journal receiver found logically damaged.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF83DE E	No API commitment resource associated with journal &2.
CPF83D1 E	Commit cycle identifier not available.
CPF8350 E	Commitment definition not found.
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.
CPF9809 E	Library &1 cannot be accessed.
CPF9810 E	Library &1 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9815 E	Member &5 file &2 in library &3 not found.
CPF9820 E	Not authorized to use library &1.
CPF9822 E	Not authorized to file &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V3R1

Top | "Journal and Commit APIs," on page 1 | APIs by category

---

Start Journal (QjoStartJournal) API

Required Parameter Group:

1	Object entry array	Input	Char(*)

2	File ID entry array	Input	Char(*)
3	Journal	Input	Char(*)

Omissible Parameter Group:

4	Start journal options	Input	Char(*)
5	Error code	I/O	Char(*)

Service Program Name: QJOURNAL
Default Public Authority: *USE
Threadsafe: Yes

The Start Journal (QjoStartJournal) API is used to start journaling changes (made to an object or list of objects) to a specific journal. The object types that are supported through this API are Data Areas (*DTAARA), Data Queues (*DTAQ), Stream Files (*STMF), Directories (*DIR), and Symbolic Links (*SYMLNK). For objects of type *STMF, *DIR, or *SYMLNK, only objects in the "root" (/), QOpenSys, and user-defined file systems are supported. For more information about the possible journal entries that can be sent, see the Journal management topic collection.

After journaling begins for the object, the user should save the journaled objects. The objects must be saved because, for example, journaled changes cannot be applied to a version of the object that was saved before journaling was in effect.

Note: For other ways to start journaling, see the following CL commands:

- Integrated File System objects - Start Journal (STRJRN)
- Access Paths - Start Journal Access Path (STRJRNAP)
- Data Areas and Data Queues - Start Journal Object (STRJRNOBJ)
- Physical Files - Start Journal Physical File (STRJRNPF)
- Libraries - Start Journal Library (STRJRNLIB)

Restrictions:

1. The object must not be journaling changes to another journal.
2. The maximum number of objects that can be associated with one journal is 250,000 or 10,000,000. To get 10,000,000, the value of *MAX10M must have been specified for the JRNOBJLMT parameter on either the Create Journal (CRTJRN) command or on the Change Journal (CHGJRN) command. This maximum number includes objects whose changes are currently being journaled, objects for which journaling was ended while the current receiver was attached, and journal receivers that were associated with the journal while the current receiver was attached. Once the number of objects is greater than or equal to this maximum, journaling does not start for any more objects.
3. The specified journal must be a local journal. Although all object types which can be journaled to a local journal can also have their changes sent to a remote journal, this is accomplished by a two step process. First start journaling to the local journal. Then connect the local journal to a remote instance. To initiate such a connection, use the Add Remote Journal (ADDRMTJRN) command or the "Add Remote Journal (QjoAddRemoteJournal) API" on page 12. For information about remote journaling, see the Journal management topic collection.
4. The specified journal and object must reside in the same auxiliary storage pool (ASP).
5. Stream files that are currently memory mapped are virtual volume files or are being used as IXS network storage spaces cannot be journaled.
6. Objects that are internally marked as not eligible for journaling cannot be journaled. The system may mark system working directories that are created inside of user directories as not eligible for journaling.
7. For data areas, only local external data area objects may be journaled. The special data areas (*LDA, *GDA, *PDA) and DDM data areas cannot be journaled.

8. For data queues, only local data queues are supported. DDM data queues cannot be journaled.
9. At least one of parameter object entry or file ID entry must not be NULL.

Authorities and Locks

Journal Authority
*OBJOPR, *OBJMGT

Non-IFS Object Authority (if specified)
*OBJOPR, *READ, *OBJMGT

IFS Object Authority (if specified)
*R, *OBJMGT (also *X if object is a directory and *ALL is specified for the directory subtree key)

Directory Authority (for each directory preceding the last component in the path name)
*X

Journal Lock
*SHRUPD

Non-IFS Object Lock (if specified)
» *EXCLRD «

IFS Object Lock (if specified)
O_RDONLY | O_SHARE_RDWR

Required Parameters

Object entry array
INPUT; CHAR(*)

The path name of the object for which changes are to be journaled.

If this parameter is NULL, the file ID entry must not be NULL.

The object entry must be in the following format.

Object Entry Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number in array
4	4	CHAR(12)	Reserved
Note: These fields repeat for each object path name.			
16	10	BINARY(4)	Length of this object path name entry
20	14	CHAR(10)	Include or omit
30	1E	CHAR(2)	Reserved
32	20	PTR(16)	Pointer to an object path name

Number in array. The number of objects in the object entry array. The possible values are 1 through 300.

Length of this object path name entry. The length of the current object path name entry that can be used as the displacement from the start of this path name entry to the next path name entry. The length must be a minimum of 32 bytes and must be a multiple of 16.

Include or omit. Whether the path name is included or omitted from the start journal operation.

**INCLUDE*	Objects that match the object name path are to be journaled, unless overridden by an *OMIT specification.
**OMIT*	Objects that match the object name path are not to be journaled. This overrides any *INCLUDE specification and is intended to be used to omit a subset of a previously selected path.

Pointer to an object path name. A pointer to the object path name for which changes are to be journaled. All relative path names are relative to the current directory at the time of the call to QjoStartJournal.

In the last component of the path name, an asterisk (*) or a question mark (?) can be used to search for patterns of names. The * tells the system to search for names that have any number of characters in the position of the * character. The ? tells the system to search for names that have a single character in the position of the ? character. Symbolic links within the path name will not be followed. If the path name begins with the tilde (~) character, then the path is assumed to be relative to the appropriate home directory.

Additional information about path name patterns is in the Integrated file system topic collection.

The pointer given points to a path name structure. If that path name structure contains a pointer, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the path name format, see Path name format.

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

File ID entry array
INPUT; CHAR(*)

The object file identifiers (FID) for which changes are to be journaled.

If this parameter is NULL, the object entry parameter must not be NULL.

The structure of this parameter follows.

Object Identifier Array Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number in array
4	4	CHAR(12)	Reserved
Note: These fields repeat for each file identifier.			
16	10	CHAR(16)	Object file identifier

Number in array. The number of objects in the object file identifier list. The possible values are 1 through 300.

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

Object file identifier. The unique 16-byte file identifier (FID) associated with integrated file system related objects.

Journal
INPUT; CHAR(*)

The path name of the journal that receives the journal entries. All relative path names are relative to the current directory at the time of the call to QjoStartJournal.

If a pointer is specified in the path name of the journal, it must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the journal path name format, see Path name format.

Omissible Parameters

Start journal options

INPUT; CHAR(*)

The start journal options, if any, to use for the selection of objects to start journaling changes and how those changes should be journaled. If this parameter is not specified, objects will be journaled using the default actions described in the field descriptions of the valid keys. See "Keys" on page 156 for the list of valid keys.

This parameter must be specified, but may be a NULL pointer.

You may specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each option record must be 16-byte aligned. If not, unpredictable results may occur.

The information must be in the following format:

Journal Options Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number of option records
4	4	CHAR(12)	Reserved
Note: These fields repeat for each option record.			
16	10	BINARY(4)	Length of option record
20	14	BINARY(4)	Key
24	18	BINARY(4)	Length of data
28	1C	CHAR(4)	Reserved
32	20	CHAR(*)	Data

Number of option records. The total number of all option records. If this field is zero, an error will be returned.

Length of option record. The length of the option record. This length is used to calculate the starting position of the next option record. If you specify a length of option record that is not equal to the key field's required option record length, an error message will be returned.

Key. Specific action for start journal. See "Keys" on page 156 for the list of valid keys.

Length of data. The length of the option record. This length is used to calculate the ending position of the data for this option.

If you specify a length of data that is not equal to the key field's required data length, an error message will be returned.

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

Data. The data that is used to determine the journal option. All values are validity checked.

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.

Keys

The following table lists the valid keys for the key field area of the journal options record. For detailed descriptions of the keys, see the "Field Descriptions."

Some messages for this API refer to parameters and values for the Start Journal (STRJRN) command. This table also can be used to locate the key names that correspond to the STRJRN command parameters.

Key	Input Type	Field	Length of Option Record	Length of Data	STRJRN Command Parameter
1	CHAR(5)	Directory subtree	32	5	SUBTREE
2	» CHAR(*)	Name pattern	*	* «	PATTERN
3	CHAR(4)	New objects inherit journaling	32	4	INHERIT
4	CHAR(6)	Images	32	6	IMAGES
5	CHAR(10)	Omit journal entry	32	10	OMTJRNE
» 6	CHAR(1)	Logging Level	32	1	LOGLVL «

Field Descriptions

Directory subtree. Whether the directory subtrees are included in the start journal operation. The default is *NONE.

Note: This parameter is ignored if the object entry parameter is not specified or if the object is not a directory.

**NONE* Only the objects that match the selection criteria are processed. The objects within selected directories are not processed implicitly.

**ALL* All objects that meet the selection criteria are processed in addition to the entire subtree of each directory that matches the selection criteria. The subtree includes all subdirectories and the objects within those subdirectories.

Images. The kinds of images that are written to the journal receiver for updates to objects. The value *BOTH is only supported for objects of type *DTAARA.

If the images parameter is not specified, the default value will be *AFTER.

**AFTER* Only *after* images are generated for changes to the objects.

**BOTH* The system generates both *before* and *after* images for changes to the objects.

» **Logging level.** Specifies the error logging level used. This key is used to determine which messages will be sent. If this key is not specified, the default is 2.

1 *ALL - The API sends all the messages that would be sent with *ERRORS and it will also send the successful completion message for each object.

2 *ERRORS - All diagnostic and escape messages are sent but the API will not send successful completion messages for each object. At the completion of this API, one completion message will be sent. «

Name pattern. The patterns to be used to include or omit objects for the start journal operation. The default will be to include all patterns that match. For the format of this field, see "Name Pattern Format."

Additional information about path name patterns is in the Integrated file system topic collection.

Note: This parameter is ignored if the object entry parameter is not specified.

Note: This parameter only applies to objects that exist when the API is called. This parameter does not apply to objects that will be created later in a journaled directory where new objects inherit journaling.

New objects inherit journaling. Whether new objects created in an object can inherit the journaling options and the journal state of the parent directory. If the new objects inherit journaling parameter is not specified, the default will be to not inherit journaling options and the journal state of the parent directory.

**NO*	New objects created within a directory will not inherit the journaling options and journal state of the parent directory.
**YES*	New objects created within a directory will inherit the journaling options and journal state of the parent directory.

Omit journal entry. The journal entries that are omitted. This parameter only supports objects of type *STMF, *DIR, or *SYMLNK that are in the "root" (/), QOpenSys, and user-defined file systems. If the omit journal entry parameter is not specified, the default will be *NONE.

**NONE*	No entries are omitted.
**OPNCLOSYN*	Open, close, and force operations on the specified objects do not generate open, close, and force journal entries. This prevents the use of TOJOBO and TOJOBC entries on the Apply Journal Changes (APYJRNCHG) command, but it saves some storage space in the journal receivers.

Name Pattern Format

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Number in array
8	8	CHAR(12)	Reserved
Note: These fields repeat for each name pattern.			
16	10	BINARY(4)	Length of this pattern entry
20	14	CHAR(10)	Include or omit
30	1E	CHAR(2)	Reserved
32	20	PTR(16)	Pointer to pattern path structure

Include or omit. Whether the name pattern is included or omitted from the start journal operation.

**INCLUDE*	Objects that match the object name pattern are to be journaled, unless overridden by an *OMIT specification.
**OMIT*	Objects that match the object name pattern are not to be journaled. This overrides an *INCLUDE specification and is intended to be used to omit a subset of a previously selected pattern.

Length of this pattern entry. The length of this pattern entry. It is used to calculate the position of the next pattern entry. This must be set to 32.

Number in array. The number of patterns in the pattern list. The possible values are 1 through 20.

Pointer to pattern path structure. A pointer to a path structure.

This pointer must be 16-byte aligned. If not, unpredictable results may occur.

For more information on the pattern path name format, see Path name format.

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

Error Messages

The following messages may be sent from this API:

Message ID	Error Message Text
CPFA0D4 E	File system error occurred.
CPF0B3F E	The reserved area is not set to binary zeros.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF694A E	Number in array value &1 for key &2 not valid.
CPF694B E	Length &1 of variable record for key &2 not valid.
CPF694C E	Variable length record data for key &1 not valid.
CPF6979 E	Journal is unusable.
CPF700A E	&1 of &2 objects started journaling.
CPF70EF E	Parameters cannot be used together.
CPF705A E	Operation failed due to remote journal.
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.
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.

Example

The following example starts journaling a directory object and all objects within that directory subtree. Additionally, it starts journaling on another object identified by its file ID.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 180.

```
#include <string.h>
#include <qjournal.h>

void main()
{
    Qjo_Object_Entry_Array_t objectEntryArray;
    Qjo_File_ID_Entry_Array_t fileIDEntryArray;

    struct PathNameStruct
    {
        Qlg_Path_Name_T header;
        char p[50];
    };

    struct PathNameStruct path;
    struct PathNameStruct journalPath;

    char pathName[] = "/CustomerData";
```

```
char jrnPathName[] = "/QSYS.LIB/ADMIN.LIB/CUSTDATA.JRN";

Qp0lFID_t fileID;

struct JournalOptionsStruct
{
    Qjo_Journal_Options_t     opts;
    char spaceForAdditionalOptions[200];
};

struct JournalOptionsStruct journalOptions;
Qjo_Option_t *optionP;

Qus_EC_t                  errorCode;


/* Setup the object's path name structure.                */
memset(&path, 0, sizeof(path));
path.header.CCSID = 37;
memcpy(path.header.Country_ID,"US",2);
memcpy(path.header.Language_ID,"ENU",3);
path.header.Path_Type = QLG_CHAR_SINGLE;
path.header.Path_Length = strlen(pathName);
path.header.Path_Name_Delimiter[0] = '/';
memcpy(path.p, pathName, path.header.Path_Length);

/* Setup the object entry array.                          */
memset(&objectEntryArray,0,sizeof(objectEntryArray));
objectEntryArray.Number_In_Array = 1;

objectEntryArray.Entry[0].Length_Of_Object_Entry =
  sizeof(objectEntryArray.Entry[0]);
memcpy(objectEntryArray.Entry[0].Include_Or_Omit,
       QJO_INC_ENT_INCLUDE,
       sizeof(objectEntryArray.Entry[0].Include_Or_Omit));
objectEntryArray.Entry[0].Path_Name =
  (Qlg_Path_Name_T *)&path;

/* Get an object's file ID.
   This example is not including the retrieval of the
   file ID for an object.  The user can see the
   Qp0lGetAttr API for information on retrieving an
   object's file ID.  This example will proceed as if the
   fileID variable is set to a valid file ID.            */

/* Setup the file ID entry array.                         */
memset(&fileIDEntryArray,0,sizeof(fileIDEntryArray));
fileIDEntryArray.Number_In_Array = 1;
memcpy(&fileIDEntryArray.Entry,
       fileID,
       sizeof(fileIDEntryArray.Entry));


/* Setup the journal's path name structure.               */
memset(&journalPath, 0, sizeof(journalPath));
journalPath.header.CCSID = 37;
memcpy(journalPath.header.Country_ID,"US",2);
memcpy(journalPath.header.Language_ID,"ENU",3);
journalPath.header.Path_Type = QLG_CHAR_SINGLE;
journalPath.header.Path_Length = strlen(jrnPathName);
journalPath.header.Path_Name_Delimiter[0] = '/';
memcpy(journalPath.p,
       jrnPathName,
       journalPath.header.Path_Length);

/* Set the journal options.                               */
memset(&journalOptions,0,sizeof(journalOptions));
```

```
    journalOptions.opts.Number_Of_Options = 3;

    /* Set the *AFTER images key.                            */
    optionP = (Qjo_Option_t *)&journalOptions.opts.Option[0];

    optionP->Length_Of_Record = QJO_KEY_MINIMUM_RECORD_LENGTH;
    optionP->Key = QJO_KEY_IMAGES;
    optionP->Length_Of_Data = QJO_KEY_IMAGES_LENGTH;
    memcpy(optionP->Data,
           QJO_IMAGES_AFTER,
           QJO_KEY_IMAGES_LENGTH);

    /* Set the inherit directory journaling attributes key.  */
    optionP = (Qjo_Option_t *)((char *)optionP +
                               optionP->Length_Of_Record);

    optionP->Length_Of_Record = QJO_KEY_MINIMUM_RECORD_LENGTH;
    optionP->Key = QJO_KEY_INHERIT;
    optionP->Length_Of_Data = QJO_KEY_INHERIT_LENGTH;
    memcpy(optionP->Data,
           QJO_INHERIT_YES,
           QJO_KEY_INHERIT_LENGTH);

    /* Set the subtree processing images key.                */
    optionP = (Qjo_Option_t *)((char *)optionP +
                               optionP->Length_Of_Record);

    optionP->Length_Of_Record = QJO_KEY_MINIMUM_RECORD_LENGTH;
    optionP->Key = QJO_KEY_SUBTREE;
    optionP->Length_Of_Data = QJO_KEY_SUBTREE_LENGTH;
    memcpy(optionP->Data,
           QJO_SUBTREE_ALL,
           QJO_KEY_SUBTREE_LENGTH);

    /* Setup the error code structure to cause an exception
       to be sent upon error.                                */
    memset(&errorCode,0,sizeof(errorCode));
    errorCode.Bytes_Provided = 0;

    QjoStartJournal(&objectEntryArray,
                    &fileIDEntryArray,
                    (Qlg_Path_Name_T *)&journalPath,
                    (Qjo_Journal_Options_t *)&journalOptions,
                    &errorCode);
}
```

API introduced: V5R1

Top | "Journal and Commit APIs," on page 1 | APIs by category

Exit Programs

These are the Exit Programs for this category.

Change Journal Receiver Exit Program

Required Parameter Group:

1	Change journal receiver exit information	Input	Char(*)

QSYSINC Member Name: ECHGRCV1
Exit Point Name: QIBM_QJO_CHG_JRNRCV
Exit Point Format Names: CRCV0100

The Change Journal Receiver exit program is called when a journal receiver has been detached from a journal.

After a journal receiver is detached, the operating system calls the user-written exit programs through the registration facility.

The exit point supports an unlimited number of exit programs. For information about adding an exit program to an exit point, see the Registration Facility.

Notes:

- Any error messages returned by the Change Journal Receiver exit program will be ignored.
- The exit programs will not be called after the detach of a remote journal receiver if the exit program only exists on the local journal system. The exit programs must exist on the same system as the journal receiver being detached to be called.
- When system-managed change journals happen during IPL or vary on processing, the registered exit programs will not be notified until some time after the IPL or vary on completes.

Restrictions

- When a user specifies JRNRCV(*GEN) or JRNRCV(jrnrcvlib/jrnrcvname) as a parameter of a CHGJRN (Change Journal) command, or issues a DLTJRN (Delete Journal) command, the system will detach the journal receiver that is currently attached to the journal. After the CHGJRN or DLTJRN processing is complete, the system will call all of the exit programs registered on the QIBM_QJO_CHG_JRNRCV exit point. The registered exit programs will be called from a system job, and because of that they will only be allowed to run for 30 seconds. If the exit program has not completed in that time, the system cancels the call.
  Notes:
  - Since the exit programs are only allowed to run for 30 seconds, we recommend that the exit program either record the journal receiver information for later processing, or submit a new job for any work that will take longer than 30 seconds.
  - Since these attempts do occur in system jobs, we recommend that the exit program not send any diagnostic, informational, or completion messages to the job log because those messages would only be in the system job logs.
- During the call to the exit programs, the debug functions, accessed via Start Debug (STRDBG), are not available to help debug any exit program problems.
- During the call to the exit programs the ASP group associated with the job will not be able to be changed. The ASP group associated with the job will be the ASP group associated with the journal receiver that was detached.
- The exit program must exist in the system ASP or in a basic user ASP. It cannot exist in an independent ASP. Any ASP group could be associated with the job when the exit program is called.

Authorities and Locks

User Profile Authority

*ALLOBJ and *SECADM to add exit programs to the registration facility

*ALLOBJ and *SECADM to remove exit programs from the registration facility

Program Data

When you register the exit program, the following program data can be optionally provided. This program data specifies the user profile under which the exit program being registered will run. If the program data is not provided, the exit programs will run under the QUSER user profile. If the user profile specified is not found on the system, the exit programs will run under the QUSER user profile.

Offset		Type	Field
Dec	Hex		
0	0	Char(10)	User profile

Required Parameter Group

Change journal receiver exit information
INPUT; CHAR(*)

Information that is needed by the exit program for notification of any change journal receiver operations. For details, see "Format of Change Journal Receiver Exit Information."

Format of Change Journal Receiver Exit Information

The following table shows the structure of the change journal receiver exit information for exit point format CRCV0100. For a description of the fields in this format, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Change journal receiver exit information length
4	4	CHAR(20)	Exit point name
24	18	CHAR(8)	Exit point format name
32	20	CHAR(10)	Journal receiver name
42	2A	CHAR(10)	Journal receiver library name
52	34	CHAR(10)	Journal name
62	3E	CHAR(10)	Journal library name
72	48	CHAR(10)	ASP device name
82	52	CHAR(1)	Journal type
83	53	CHAR(1)	Remote journal type
84	54	CHAR(13)	Detached date and time

Field Descriptions

ASP device name. The name of the independent auxiliary storage pool (ASP) device on which the journal receiver resides. *SYSBAS is used to indicate the system ASP and all basic user ASPs.

Change journal receiver exit information length. The length in bytes of all data passed to the change journal receiver exit program.

Detached date and time. The date and time that this journal receiver was detached from the journal. For a journal receiver that was attached to a *REMOTE journal, this is the date and time that the journal receiver was detached on the local system. This field is in the CYYMMDDHHMMSS format as follows:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hours
MM	Minutes

SS Seconds

Exit point format name. The format name for the change journal receiver exit program. The possible format name follows:

CRCV0100 The format name that is used after a user journal receiver is detached.

Exit point name. The name of the exit point that is calling the exit program.

Journal library name. The library name of the journal that is associated with the journal receiver being detached.

Journal name. The name of the journal that is associated with the journal receiver being detached.

Journal receiver library name. The library name of the journal receiver being detached.

Journal receiver name. The name of the journal receiver being detached.

Journal type. An indication of whether the journal currently associated with the journal receiver being detached is local or remote. The possible values are:

0 *LOCAL
1 *REMOTE

Remote journal type. If this journal receiver was attached to a remote journal, this field is the remote journal type for that journal, when this journal receiver was attached. The possible values are:

0 The journal receiver was not attached to a remote journal. It was attached to a local journal.
1 The journal receiver was attached to a *TYPE1 remote journal.
2 The journal receiver was attached to a *TYPE2 remote journal.

User profile. The exit program will be called under this user profile. If the user profile is not found at the time the exit programs are called, the QUSER user profile will be used.

Usage Notes

1. If multiple exit programs will be interested in the same journal and journal receivers, it is recommended that the order in which the exit programs will be called be considered prior to registering the exit programs. For example, two exit programs are both interested in a particular journal receiver. Exit program 1 deletes the journal receiver and exit program 2 saves the journal receiver. Exit program 2 will fail because the journal receiver is no longer on the system after exit program 1 is called. Exit program 2 must run before exit program 1.
2. If an exit program times out or signals an error, no more exit programs are called. For this reason, it is highly recommended that the exit programs submit jobs to perform the necessary functions, so that all exit programs will be called.
3. Any journal whose associated journal receivers are being monitored by an exit program should have the Delete Receiver (DLTRCV) option set to *NO.

« Exit program introduced: V6R1

Top | "Journal and Commit APIs," on page 1 | APIs by category

Commitment Control Exit Program

Required Parameter Group:

1	Commitment control exit program information	Input	Char(80)
2	Status information	Input	Char(*)

Optional Parameter:

3	Return information	Output	Char(*)

QSYSINC Member Name: ETNCMTRB

Users who add API commitment resources to the commitment definition must supply a commitment control exit program as described in Qualified commitment control exit program name (page 5). The commitment control operations call this exit program:

- Optionally when the commitment definition associated with this resource is placed in a rollback-required state.
- Optionally during the classify phase of commit or rollback processing.
- Optionally during the prepare phase of commit processing.
- To commit during commit processing.
- To roll back during rollback processing.
- Optionally to reacquire locks during IPL or ASP device vary on.

The commitment control operations pass specific information to the commitment control exit program. The exit program must be coded to handle this specific information as described in "Required Parameter Group."

Authorities and Locks

None.

Required Parameter Group

Commitment control exit program information
INPUT; CHAR(80)

Information associated with the commitment control exit program specified when the API commitment resource was added to the commitment definition. This information is passed to the exit program exactly as it was entered when the API commitment resource was added. The area may contain any data such as pointers or an object name. If pointers are used, each one must start on a 16-byte boundary. A pointer may refer to an area of storage that contains information required by your exit program. A pointer may refer only to an area of storage on an ASP that is available when the exit program is called.

Status information
INPUT; CHAR(*)

Status information from the commitment control operations. Each field of this information has a specific meaning. The fields, their meanings, and size are shown in "Status Information Format" on page 165.

Optional Parameter

Return information
OUTPUT; CHAR(*)

Information returned from the commitment control exit program. Each field of this information has a specific meaning. The fields, their meanings, and size are shown in "Return Information Format" on page 168.

This parameter is not passed to the commitment control exit program if the Add resource options parameter was not coded on the Add Commitment Resource (QTNADDCR) API when the resource was registered.

Status Information Format

The following table shows the offsets, type, and name for the fields passed to the exit program as status information. See "Field Descriptions" for a description of each of these fields.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Status information length
4	4	CHAR(1)	Action required
5	5	CHAR(1)	Called for IPL recovery or ASP device vary on
6	6	CHAR(4)	Reserved
10	A	CHAR(1)	Process error status
11	B	CHAR(1)	Process end status
12	C	CHAR(1)	Reserved
13	D	CHAR(1)	Commit or rollback qualifier
14	E	CHAR(1)	Commitment definition scope
15	F	CHAR(25)	Reserved
40	28	BINARY(4)	Commit cycle identifier
44	2C	CHAR(10)	Journal name
54	36	CHAR(10)	Journal library name
64	40	CHAR(39)	Logical unit of work identifier
103	67	CHAR(1)	Commitment definition status
104	68	Binary(4)	Active savepoint
108	6C	Binary(4)	Savepoint number
112	70	CHAR(20)	Commit cycle identifier - long
132	84	Char(4)	Reserved
136	88	Char(128)	Savepoint name

Field Descriptions

Action required. The action the commitment control exit program is called to perform. The possible values are:

A Exit program called as a last agent. The program owns the decision of whether the logical unit of work is committed or rolled back. The program must commit or rollback its resources and inform the system of the decision with the Commit vote field of the Return information parameter.
B Exit program called to place its resources in a rollback-required state.
C Exit program called to commit its resources.
E Exit program called to set a savepoint in its resources.
F Exit program called to release a savepoint in its resources.
G Exit program called to rollback its resources to a savepoint.

L Exit program called to reacquire its locks. This happens when the status of the API commitment resource is found to be in doubt during an IPL or ASP device vary on. The locks should be released when the exit program is called to commit or rollback its resources after the IPL or vary on completes.
P Exit program called to prepare its resources.
R Exit program called to rollback its resources.
S Exit program called to classify its resources prior to a rollback operation.
Y Exit program called to classify its resources prior to a commit operation.

Note: The commitment control exit program is called for actions A, B, L, P, S, and Y only if it is indicated when the resource was added that calls should be made to do these actions.

Active savepoint. The identifier assigned to the savepoint that was active when the commit, set savepoint, release savepoint, rollback to savepoint or rollback was requested. A value of 1 indicates there were no active savepoints. This identifier may not increment by 1 for consecutive savepoints because of savepoints created internally by the system. This value applies only when the Action required field is C, E, F, G or R.

Called for IPL or ASP device vary on recovery. Whether the exit program was called to perform IPL or ASP device vary on recovery processing for the API commitment resource. The possible values are:

N Not called to perform IPL or ASP device vary on recovery processing for the API commitment resource.The action required field may have any valid value.
P Called to perform recovery for the API commitment resource after the IPL or ASP device vary on is completed. The purpose of this call is to commit or rollback resources whose status was found to be in doubt during the IPL or ASP device vary on. These resources were called to reacquire locks during the IPL or ASP device vary on if so indicated when the resource was added. The action required field will be C (exit program called to commit its resources) or R (exit program called to rollback its resources).
Y Called during IPL recovery processing for the API commitment resource. The action required field will be C (exit program called to commit its resources), L (exit program called to reacquire its locks), or R (exit program called to rollback its resources).

Commit cycle identifier. Commit cycle identifier of the current commit cycle. This value is provided only if a journal name was specified when the API resource associated with the exit program was added. If no journal name was specified this field will be zero.

This commit cycle identifier applies only to the journal specified when the resource was added. If the journal has been placed in STANDBY state, this field will be zero.

This field will be -1 if the value could not fit in the specified Binary(4) field. The complete value will always be provided in the Commit cycle identifier - long field.

Commit cycle identifier - long. The same field as Commit cycle identifier except the information is in a Char(20) field that is treated as Zoned(20,0).

Commit or rollback qualifier. If the commit or rollback operation is being performed on behalf of an explicit request by a program or is being performed implicitly by the system.

E Explicit commit or rollback (initiated by the user)
I Implicit commit or rollback (initiated by the system)

This commit or rollback qualifier applies only when the action required is C, E, F, G, P, R, S, or Y. For all other actions, a blank is sent.

Commitment definition scope. The scope for the commitment definition. The possible values are:

A Activation group level
J Job level

Commitment definition status. The overall status of the commitment definition currently active for the activation group for the program performing the retrieve request. The scope for this commitment definition is returned in the commitment definition scope field. The possible values are:

L The commitment definition is active on the local system within the activation group for the program performing the retrieve request. An *L* is returned if one or more of the following resources are under commitment control.
- Local, open database files
- Local, closed database files with pending changes
- Resources with object-level changes
- Local relational database resources
- API commitment resources

B The commitment definition is active on both the local and one or more remote systems.

Journal library name. The journal library name specified when the commitment resource was added to the commitment definition. If *CURLIB or *LIBL was specified for the library when the resource was added, the actual library name at the time the resource was added is placed in this field. If no journal was specified when the resource was added, blanks are placed in this field.

Journal name. The journal name specified when the commitment resource was added to the commitment definition. If no journal was specified when the resource was added, a value of *NONE is placed in this field.

Logical unit of work identifier. The identifier for the logical unit of work currently associated with this commitment definition.

Logical Unit of Work Identifier Format		
Field	**Type**	**Description**
Network ID	CHAR(0-8)	Network identifier
Separator	CHAR(1)	The separator character ".”
Local location name	CHAR(0-8)	The name of the local location
Separator	CHAR(3)	The separator characters ".X'"
Instance number	CHAR(12)	The hex value of the instance number converted to decimal
Separator	CHAR(2)	The separator characters "'."
Sequence number	CHAR(5)	The hex value of the sequence number converted to decimal

Process end status. If the exit program was called because of process end, and if so, how the process is ending, or if the exit program was called as the result of an activation group ending. The possible values are:

0 Not during the process or activation group end
1 Normal process end; job ended with a zero completion code
2 Abnormal process end; job ended with a completion code that is not zero
4 Activation group is ending

Process error status. If errors occurred in the commitment control processing for this logical unit of work prior to this call to the exit program. The possible values are:

0 No errors occurred
1 Errors occurred

Reserved. An ignored field.

Savepoint number. The identifier assigned to the savepoint that is being set, released or rolled back. This identifier may not increment by 1 for consecutive savepoints because of savepoints created internally by the system. This value applies only when the Action required field is E, F or G.

Savepoint name. The name that identifies the savepoint that is being set, released or rolled back. This value applies only when the Action required field is E, F or G.

Status information length. The length in bytes of all data passed to the Commitment control exit program.

Return Information Format

The following table shows the offsets, type, and name for the fields returned from the exit program.

Offset		Type	Field
Dec	**Hex**		
0	0	Binary(4)	Return information length
4	4	Char(1)	Commit vote
5	5	Char(1)	Classify result
6	6	Char(1)	Changes ended

Field Descriptions

Changes ended. This field is used when the commitment control exit program is called with the Action required field set to A, C, E, F, G or R. It determines whether the commitment resource should be removed at the end of the commit or rollback operation. The possible values are:

0 The resource should not be removed at the end of the commit or rollback operation.
1 The resource should be unconditionally removed at the end of the commit or rollback operation.
2 The resource should be removed only if the commit operation was successful. If the commit operation is not successful the resource is not removed and the Changes Ended field is set back to 0.

If a valid value is not returned, message CPD835E is issued and the resource is not removed.

Classify result. This field is used when the commitment control exit program is called with the Action required field set to S or Y. The possible values are:

0 The classify was successful.
1 The classify was not successful. The commit or rollback operation is ended and message CPF835F is issued. A recoverable failure should be reported for this resource.

If a valid value is not returned, message CPD835E is issued and the classify is considered unsuccessful.

Commit vote. This field is used when the commitment control exit program is called with the Action required field set to A or P. At this point the exit program has a chance to vote whether the entire logical unit of work should commit or roll back. If the exit program votes to roll back, the logical unit of work will roll back regardless of any other votes.

The exit program can also vote read-only. This tells the system that this resource has had no changes made to it and it does not matter if the logical unit of work commits or rolls back. If this exit program votes read-only, it will not be called to commit or roll back this logical unit of work. The possible values are:

1 The commitment control exit program votes to commit the logical unit of work.
2 The commitment control exit program votes to roll back the logical unit of work.
3 The commitment control exit program votes read-only and does not want a call to commit or roll back this logical unit of work.

If a valid value is not returned, message CPD83DE is issued and the logical unit of work is rolled back.

Return Information Length. The length in bytes of all data returned from the commitment control exit program. This field is used to determine whether a particular return value should be used or not. The only valid value for this field is 7. If the returned length is not 7, message CPD83DE is issued and all the other return fields are considered to be not valid.

Exit Program Locks

Commitment control obtains a shared-no-update (*SHRNUP) lock on the exit program when the commitment resource is added using the Add Commitment Resource (QTNADDCR) API. This lock is maintained until the resource is removed using the Remove Commitment Resource (QTNRMVCR) API. This locking is done to prevent any changes by other processes to the Commitment control exit program. Changes by other processes, such as deletion, modification, or authority changes, are prevented.

Exit Program Coding Guidelines

When coding a commitment control exit program, consider the items in the following lists.

Your exit program **must**:

- Complete its processing within 5 minutes. During process end or IPL, or ASP device vary on recovery processing, the system does not allow a Commitment control exit program to run more than 5 minutes. An exit program will not be allowed to prevent a process from ending or an IPL or ASP device vary on from completing.
- Return an exception to a commitment control operation only if there has been a failure in the exit program. If the exit program signals an escape message to commitment control, the system assumes there is a failure. A diagnostic message with a final escape message is returned to the calling program, or a message is sent to the system operator if the error occurs during or after IPL or ASP device vary on processing.
- Perform any necessary cleanup of locks acquired by the exit program. This is especially important when the program is called after IPL or ASP device vary on recovery to commit or rollback resources whose statuses were found to be in doubt and were called to reacquire locks during IPL or ASP device vary on recovery.
- Be written expecting to be called as part of every commitment control operation that is performed for a commitment definition, including implicit commitment control operations performed by the system at:
  - Activation group end
  - Job end
  - IPL or ASP device vary on time (optionally)
- Be threadsafe if the API commitment resource is added in a multithreaded job.

Your exit program **must not** perform any of these operations if the scope for the commitment definition is the job level, or any of these functions from the same activation group if the scope for the commitment definition is the activation group level.

- Call any commit or rollback operations such as the CL COMMIT command or SQL COMMIT statement. If it does, message CPF8367 is returned to the exit program.
- Call the QTNADDCR, the QTNRMVCR, or the QTNRBRQD API. If it does, message CPF8367 is returned to the exit program.
- Open a local database or DDM file member under commitment control. If it does, message CPF432A is returned to the exit program.
- Start commitment control. If it does, message CPF8351 is returned to the exit program.
- End commitment control. If it does, message CPF8367 is returned to the exit program.
- Use any protected conversations. If it does, a return code is returned to the exit program.
- Connect to a remote relational database with a program that is running under commitment control. If it does, either a return code or an error message is returned to the exit program.

Your exit program **should not** attempt any of these functions if the scope for the commitment definition is the job level, or any of these functions from the same activation group if the scope for the commitment definition is the activation group level.

- Record-level I/O for a local database or DDM file member opened under commitment control
- SQL statements under commitment control

If either of these functions are performed, the results are unpredictable and no error messages are issued.

The following items are good guidelines to follow for any program you write. Your program **should**:

- Handle all potential error conditions (fault tolerant). Perform any necessary cleanup of locks acquired by the exit program.
- Prevent the potential for any infinite looping conditions. The system stops the exit program, after 5 minutes, during process end, IPL or ASP device vary on time.
- Be relatively short and perform well.
- Be callable during IPL or ASP device vary on to reacquire locks and to recover resources.
- Notify the application when placing a commitment definition in rollback-required state.
- Release all locks before finishing IPL or ASP device vary on recovery.

If your exit program changes any of the required parameter values passed to it, these changes are not preserved for future calls to the exit program.

Process End, Activation Group End, and IPL or ASP Device Vary On Recovery Processing Guidelines

During process end, activation group end, and IPL or ASP device vary on recovery processing, the debug functions are not available to help debug any exit program problems. The following operations may be performed during these processing phases. If any other actions take place, the Commitment control exit program may not run successfully or the results will be unpredictable.

- Working with physical files, including creating, changing, opening, closing, clearing, and deleting
- Database input and output operations
- Working with data areas, including creation, changing, retrieving, and deletion
- Working with data queues, including creation and deletion
- Working with message queues, including creation, clearing, changing and deletion

Some examples of things your exit program might not be able to do during process end, activation group end, IPL, or ASP device vary on are:

- Signal any inquiry messages.
- Submit any other jobs.
- Use or attempt to start any remote communications activities.

- Start any subsystems.
- Include a commit cycle identifier if sending journal entries using the Send Journal Entry (QJOSJRNE) API. This restriction applies during IPL only.

When called after IPL or ASP device vary on recovery to commit or rollback resources whose status was found to be in doubt during IPL or ASP device vary on recovery, the exit program will be called in a system database server job. The job name for these jobs The job name for these jobs on the system ASP start with the characters QDBSRV and end with a number beginning with 02 (for example, QDBSRV02, QDBSRV03, and so forth). On an IASP, the job name for these jobs starts with the characters "QDBS" followed by three digits of the ASP device number and ends with the character "V" and a number beginning with 02 (for example for ASP device number 34, QDBS034V02, QDBS034V03, and so forth). Debug functions can be used for these jobs by using the Start Service Job (STRSRVJOB) command.

Exit program introduced: V2R2. Formerly called Commit and Rollback.

Top | "Journal and Commit APIs," on page 1 | APIs by category

Delete Journal Receiver Exit Program

Required Parameter Group:

1	Delete journal receiver exit information	Input	Char(*)
2	Status information	Output	Char(*)

QSYSINC Member Name: EDLTRCV1
Exit Point Name: QIBM_QJO_DLT_JRNRCV
Exit Point Format Names: DRCV0100

The Delete Journal Receiver exit program is called when a journal receiver is to be deleted.

When a journal receiver is to be deleted, the operating system calls the user-written exit programs through the registration facility. The exit programs will be called before the journal receiver is actually deleted and can indicate whether the exit program considers the receiver eligible for deletion.

The exit point supports an unlimited number of exit programs. For information about adding an exit program to an exit point, see the Registration Facility.

Note: If the Delete Journal Receiver exit program returns any error messages, the journal receiver will not be considered eligible for deletion.

Restrictions

- When a user specifies DLTRCV(*YES) as an attribute of a journal, the system will attempt to delete the journal receiver when the system sees it is no longer required for recovery purposes. Before the journal receiver is deleted, the system will call all of the exit programs registered for the QIBM_QJO_DLT_JRNRCV exit point. If any of the exit programs give an indication that the journal receiver is not eligible for deletion, then the journal receiver will not be deleted. Instead, the system will retry the deletion in the time specified for the DLTRCVDLY value for the journal. These system deletion attempts take place in system jobs, during an IPL and during the vary on of an independent ASP; therefore, the exit program will be allowed to run for only 5 minutes when called during either of these conditions. If the exit program has not completed in that time, the system cancels the call, and the journal receiver will not be considered eligible for deletion.

  Note: Since these attempts do occur in system jobs, we recommend that the exit program not send any diagnostic, informational, or completion messages to the job log because those messages would only be in the system job logs.

- If the delete journal receiver is called as part of process end, the exit program can run for only 5 minutes. If it exceeds 5 minutes, the call is canceled, and the journal receiver is not eligible for deletion.
- During the call to the exit programs, the debug functions, accessed via Start Debug (STRDBG), are not available to help debug any exit program problems.
- During the call to the exit programs the ASP group associated with the job will not be able to be changed. The ASP group associated with the job will be the ASP group associated with the journal receiver to be deleted.
- The exit programs must exist in the system Auxiliary Storage Pool (ASP) or in a basic user ASP. It cannot exist in an independent ASP. Any ASP group could be associated with the job when the exit program is called. If the exit program is not found, the journal receiver will not be considered eligible for deletion.

Authorities and Locks

User Profile Authority

*ALLOBJ and *SECADM to add exit programs to the registration facility

*ALLOBJ and *SECADM to remove exit programs from the registration facility

Program Data

When you register the exit program, the following program data can be optionally provided. This program data specifies the user profile under which the exit program being registered will run. If the program data is not provided, the exit programs will run under the QUSER user profile.

Offset		Type	Field
Dec	Hex		
0	0	Char(10)	User profile

Required Parameter Group

Delete journal receiver exit information

INPUT; CHAR(*)

Information that is needed by the exit program for notification of any journal receiver deletions. For details, see "Format of Delete Journal Receiver Exit Information."

Status information

OUTPUT; CHAR(*)

Information that is returned by the exit program stating whether the deletion can occur or not. For details, see "Format of Status Information" on page 173.

Format of Delete Journal Receiver Exit Information

The following table shows the structure of the delete journal receiver exit information for exit point format DRCV0100. For a description of the fields in this format, see "Field Descriptions" on page 173.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Delete journal receiver exit information length
4	4	CHAR(20)	Exit point name
24	18	CHAR(8)	Exit point format name
32	20	CHAR(10)	Journal receiver name

Offset		Type	Field
Dec	Hex		
42	2A	CHAR(10)	Journal receiver library name
52	34	CHAR(10)	Journal name
62	3E	CHAR(10)	Journal library name
72	48	CHAR(1)	Called by system job
73	49	CHAR(1)	Called during IPL or vary on of an independent ASP.
74	4A	CHAR(1)	Called during process end
75	4B	CHAR(1)	Journal type
76	4C	CHAR(1)	Remote journal type
77	4D	CHAR(1)	Save status
78	4E	CHAR(1)	Partial status
79	4F	CHAR(13)	Detached date and time

Format of Status Information

The following table shows the structure of the status information. For a description of the fields in this format, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Delete status information length
4	4	CHAR(1)	Delete status

Field Descriptions

Called by system job. Whether this call was from a user job or a system job. The possible values are:

0 The call is from a user job.
1 The call is from a system job, and is therefore limited to 5 minutes.

Called during IPL or vary on of an independent ASP. Whether this call was during an IPL or during the vary on of an independent ASP. The possible values are:

0 The call is not during an IPL or vary on of an independent ASP.
1 The call is during an IPL or vary on of an independent ASP, and is therefore limited to 5 minutes.

Called during process end. Whether this call was during process end. The possible values are:

0 The call is not during process end.
1 The call is during process end.

Delete journal receiver exit information length. The length in bytes of all data passed to the delete journal receiver exit program.

Delete status. The status value that indicates whether the exit program considers the journal receiver eligible for deletion. The possible values are:

0 The journal receiver is not eligible for deletion. Therefore, the delete journal receiver command will be ended, and the receiver will not be deleted.
1 The journal receiver is eligible for deletion.

Note: If any other value is specified for this item, it will be ignored, and the journal receiver will **not** be eligible for deletion.

Delete status information length. The length in bytes of all data returned from the Delete Journal Receiver exit program. The only valid value for this field is 5. If anything else is entered, the receiver is not considered eligible for deletion.

Detached date and time. The date and time that this journal receiver was detached from the journal. For a journal receiver that was attached to a *REMOTE journal, this is the date and time that the journal receiver was detached on the local system. This field is in the CYYMMDDHHMMSS format as follows:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YYYY	Year
MM	Month
DD	Day
HH	Hours
MM	Minutes
SS	Seconds

If the journal receiver was never attached to a journal, this field will be blank. If the journal receiver was never detached from a journal, or if this journal receiver is a partial receiver, this field will be all zeros.

Exit point format name. The format name for the delete journal receiver exit program. The possible format name follows:

DRCV0100 The format name that is used before a user journal receiver is to be deleted.

Exit point name. The name of the exit point that is calling the exit program.

Journal library name. The library name of the journal that is associated with the journal receiver library being deleted. If there is no journal associated with this journal receiver, this field will be blank.

Journal name. The name of the journal that is associated with the journal receiver being deleted. If there is no journal associated with this journal receiver, this field will be blank.

Journal receiver library name. The name of the journal receiver library being deleted.

Journal receiver name. The name of the journal receiver being deleted.

Journal type. An indication of whether the journal currently associated with the journal receiver being deleted is local or remote. The possible values are:

blank The journal receiver has not yet been attached to any journal or the receiver is not currently associated with any journal.
0 *LOCAL
1 *REMOTE

Partial status. An indication of whether the journal receiver is a partial receiver. A journal receiver is partial for one of the following reasons:

- It was restored from a version that was saved while it was attached to the journal. Additional journal entries may have been written that were not restored.
- It was one of a pair of dual journal receivers, and it was found damaged while attached to the journal. The journal receiver has since been detached. This journal receiver is considered partial because additional journal entries may have been written to the dual journal receiver.
- It is associated with a remote journal and it does not contain all the journal entries that are in the associated journal receiver attached to the source journal.

The possible values are:

0 The journal receiver is not a partial journal receiver.
1 The journal receiver is a partial journal receiver.

Remote journal type. If this journal receiver was attached to a remote journal, this field is the remote journal type for that journal, when this journal receiver was attached. The possible values are:

blank The journal receiver has not yet been attached to any journal.
0 The journal receiver was attached to a local journal.
1 The journal receiver was attached to a *TYPE1 remote journal.
2 The journal receiver was attached to a *TYPE2 remote journal.

Save status. An indication of whether the journal receiver has been saved after it was detached. The possible values are:

0 The journal receiver has not been saved after it was detached.
1 The journal receiver has been saved after it was detached.

User profile. The exit program will be called under this user profile. If the user profile is not valid at the time the exit programs are called, the QUSER user profile will be used.

IPL Processing Guidelines

The following operations may be performed during the IPL. If any other actions take place, the Delete Journal Receiver exit program may not run successfully or the results will be unpredictable.

- Working with physical files, including creating, changing, opening, closing, clearing, and deleting
- Database input and output operations
- Working with data areas, including creating, changing, retrieving, and deleting
- Working with data queues, including creating and deleting
- Working with message queues, including creating, clearing, changing, and deleting

Some examples of things your exit program might not be able to do during IPL are:

- Signal any inquiry messages
- Submit any other jobs
- Use or attempt to start any remote communications activities
- Start any subsystems

Exit program introduced: V4R2

Top | "Journal and Commit APIs," on page 1 | APIs by category

Concepts

These are the concepts for this category.

Journaling for Journal and Commit APIs

Journaling allows you to specify database files or access paths you want to protect for recovery purposes, or allows you to provide an audit trail for changes to database files. In addition, journaling allows you to provide an audit or activity trail for other objects or activities either by system activities, such as security auditing, or by user activities, such as the Send Journal Entry (SNDJRNE) command or QJOSJRNE API.

Two objects are associated with journaling: journals and journal receivers.

A **journal** is the object that identifies:
- The journaled objects, if any
- The current journal receivers
- Any journal receivers on the system that have been, or are, associated with the journal
- Any remote journals that are associated with the journal

A **local journal** is the journal that is the initiator of the original journal deposit. Objects can be journaled to a local journal, and journal entries are deposited into a local journal due to changes made to the journaled objects, or because journal entries were sent to the local journal.

A **remote journal** is a journal that has been associated with another journal via the Add Remote Journal (QjoAddRemoteJournal) API. Objects cannot be journaled to a remote journal nor can journal entries be directly deposited into a remote journal. Instead, a remote journal has journal entries replicated to it from its upstream source journal. The upstream source journal can be either a local journal or a remote journal.

A **journal receiver** is the object that contains journal entries, and can be associated with either a local or remote journal. For a journal receiver attached to a local journal, journal entries are directly deposited into the journal receiver. For a journal receiver attached to a remote journal, journal entries are replicated into the journal receiver from the source journal receiver associated with the upstream source journal.

Journal entries contain information such as:
- The job name, program name, and user that caused the journal entry to be deposited into the local journal.
- The date and time the journal entry was deposited into the local journal.
- Journal code.
- Entry type.
- Information that is unique to each journal entry type, called entry specific data.

For example, the entry specific data associated with the put of a record to a physical file member (journal code R, entry type PT) contains an image of the actual record that was put. The rest of the journal entry information helps determine at what time, by what user the entry was sent, and other details. If a user sends information to the journal using the SNDJRNE command or QJOSJRNE API (journal code U, entry type is defined at send time by the user), the entry specific data contains what was specified at that time.

When journaling is started for an object, a unique identifier called a **journal identifier (JID)** is assigned. This identifier remains the same even if the object is renamed or moved. The journal identifier is associated with every journal entry that is associated with a specific journaled object.

The JID allows the journal facility to associate the current name of an object with the journal entries, even if the entry was made before the object was renamed. To determine what name is currently associated

with a particular JID, use the QJORJIDI API. See “Retrieve Journal Identifier Information (QJORJIDI) API” on page 101 (QJORJIDI) API for more information about the JID and this API.

The SNDJRNE command and the QJOSJRNE API provide similar function, the sending of a journal entry at a user’s request. See “Send Journal Entry (QJOSJRNE) API” on page 144 (QJOSJRNE) API for more information about this API. The major differences between the API and command are:

- The API allows up to 15761440 bytes of entry specific data, as opposed to 3000 bytes with the command.
- The API allows the association of the current commit cycle identifier with the journal entry. This support can be used with the QTNADDCR API. For more information see “Add Commitment Resource (QTNADDCR) API” on page 3 (QTNADDCR) API.
- The API can optionally return the journal sequence number and journal receiver information for the entry that was deposited.

The QjoAddRemoteJournal, QjoRemoveRemoteJournal, and QjoChangeJournalState APIs allow you to establish, manipulate, and maintain a remote journal environment. A remote journal environment allows you to replicate journal entries from one system to another via communications methods, such as TCP/IP, SNA, and OptiConnect for i5/OS®. This support can be used to help replicate data from one system to one or more additional systems. Using application programs, the replicated entries can then be used to maintain a backup, or replica of the primary systems data. If desired, the backup data can be used in the event the primary system fails. See “Add Remote Journal (QjoAddRemoteJournal) API” on page 12 (QjoAddRemoteJournal) API, “Change Journal State (QjoChangeJournalState) API” on page 23 (QjoChangeJournalState) API, and “Remove Remote Journal (QjoRemoveRemoteJournal) API” on page 44 (QjoRemoveRemoteJournal) API for more information about these APIs. The Add Remote Journal (ADDRMTJRN), Change Remote Journal (CHGRMTJRN), and Remove Remote Journal (RMVRMTJRN) commands provide support similar to these APIs.

The QjoRetrieveJournalInformation and QjoRtvJrnReceiverInformation APIs provide information that is similar to the Work with Journal Attributes (WRKJRNA) and Display Journal Receiver Attributes (DSPJRNCVA) commands, respectively. See “Retrieve Journal Information (QjoRetrieveJournalInformation) API” on page 105 (QjoRetrieveJournalInformation) API and “Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API” on page 131 (QjoRtvJrnReceiverInformation) API for more information about these APIs.

Top | “Journal and Commit APIs,” on page 1 | APIs by category

Commitment Control for Journal and Commit APIs

The terms *commit* and *rollback* include all methods of commit and rollback available on the system, such as:

- CL COMMIT and ROLLBACK commands
- ILE C_Rcommit and _Rrollbck functions
- SQL COMMIT and ROLLBACK statements

A **commitment resource** is any part of the system that is used by a process and placed under commitment control. When a part of the system is put under commitment control by the Add Commitment Resource (QTNADDCR) API, that resource can be referred to as an **API commitment resource**.

API commitment resources are processed by the system during:

- Commit
- Rollback
- Process end

- Activation group end
- and, optionally, during:
  - Initial program load (IPL)
  - The classify phase of a commit or rollback
  - The prepare phase of a commit

When commitment control is started using the Start Commitment Control (STRCMTCTL) command, the system creates a **commitment definition** that is scoped to a particular activation group or to the job as indicated on the commit scope (CMTSCOPE) keyword.

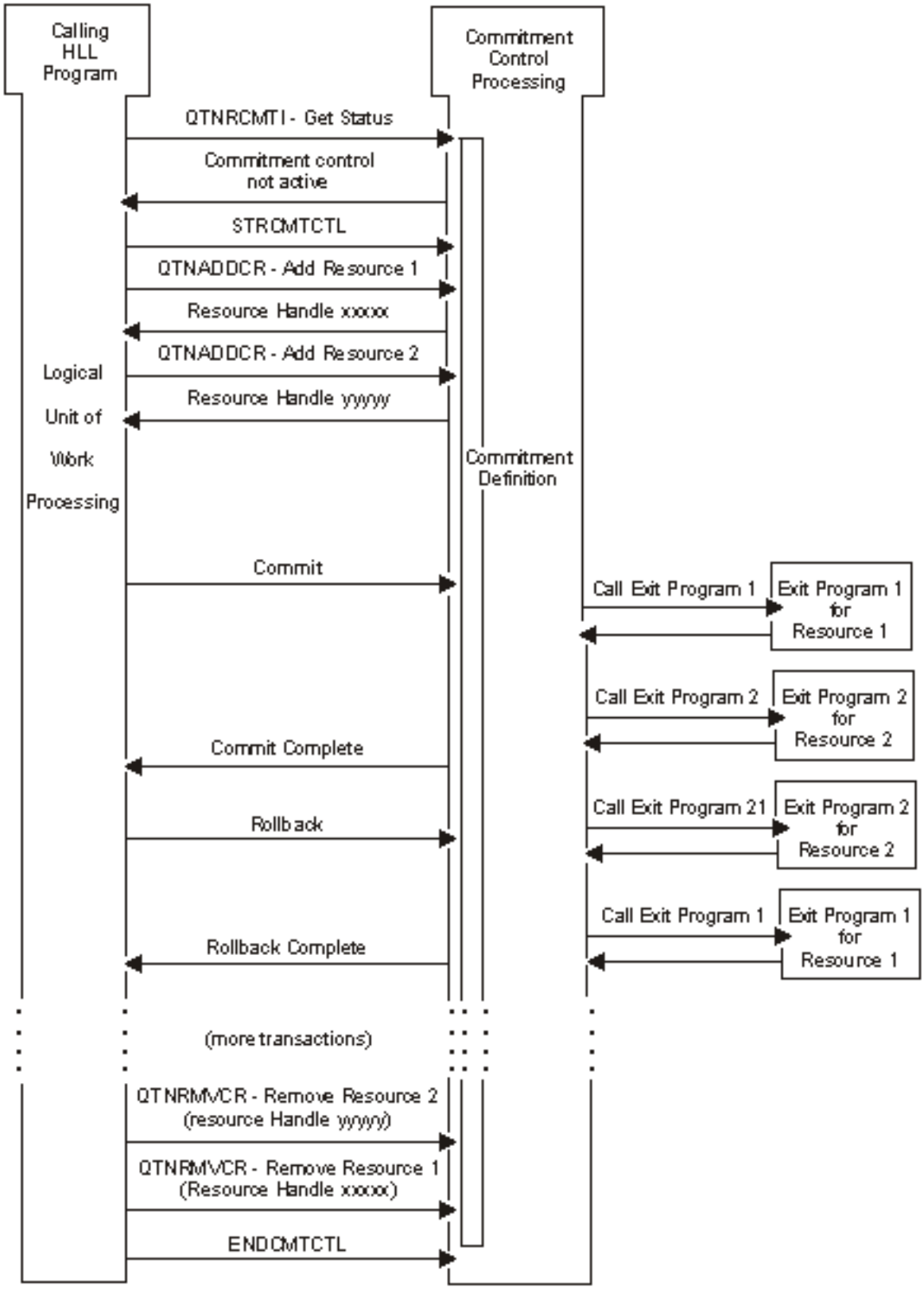
Each group of committable changes is intended to be an atomic operation. Each group can be committed (changes are made permanent to the system) or rolled back (changes are permanently removed from the system) and is referred to as a logical unit of work. The first logical unit of work begins when commitment control is started. Each commit and rollback completes the current logical unit of work and starts a new one.

A commitment definition saves internal control information pertaining to the resources under commitment control. This internal control information is maintained as the state of those commitment resources changes, until that commitment definition is ended using the End Commitment Control (ENDCMTCTL) command. A commitment definition generally includes:

- The parameters on the Start Commitment Control (STRCMTCTL) command
- The current status of the commitment definition
- Information about files and other resources that contain changes made during the current logical unit of work

The Example Using Selective Commitment Control APIs (page 178) shows how some of the commitment control APIs can be used together. First, the Retrieve Commitment Information (QTNRCMTI) API is used by the high-level language (HLL) program to determine if commitment control is active within the activation group for the HLL program. If the activation-group-level commitment definition is already active, then the status retrieved by the API will be with respect to that activation-group-level commitment definition. If the activation-group-level commitment definition is not active, but the job-level commitment definition is active, then the status retrieved by the API will be with respect to the job-level commitment definition. If status is being retrieved for the job-level commitment definition, then information from a second status field returned by the API can be used to determine whether programs that have run within the activation group have already used the job-level commitment definition. If no program running within the activation group has used the job-level commitment definition, then an activation-group-level commitment definition may be started by the HLL program.

Example Using Selective Commitment Control APIs



Note: Programs running within a single activation group may use the activation-group-level or the job-level commitment definition, but cannot use both definitions concurrently. Two programs running within different activation groups may each use a separate activation-group-level commitment definition, or one or both programs may use the job-level commitment definition.

Once a commitment definition has been established for the HLL program, the Add Commitment Resource (QTNADDCR) API is used to add one or more commitment resources to the commitment definition. When a commit or rollback operation is performed to complete a transaction for this commitment definition, the system performs the commit or rollback operation for all record-level and object-level resources. It also calls an exit program, as identified by the HLL program when the API commitment resource was added, for each API commitment resource.

The Example Using Selective Commitment Control APIs (page 178) shows one call to each exit program during commit and rollback operations. This is the case for one-phase resources. For two-phase resources, up to three calls are made during commit operations, and up to two calls are made during rollback operations. See "Add Commitment Resource (QTNADDCR) API" on page 3 (QTNADDCR) API for more information about one-phase and two-phase commitment resources.

After all the desired logical units of work are completed by the HLL program, the Remove Commitment Resource (QTNRMVCR) API must be used to remove each of the commitment resources added to the commitment definition before the commitment definition can be ended by the End Commitment Control (ENDCMTCTL) command. However, if an activation-group-level commitment definition is being used and the activation group is ended when the HLL program returns, then any API commitment resources are implicitly removed by the system and the activation-group-level commitment definition is automatically ended by the system. Prior to the system implicitly removing the API commitment resources and automatically ending the activation-group-level commitment definition, an implicit commit or rollback operation is performed by the system if pending changes exist for the commitment definition, with the appropriate exit program calls made for any API commitment resources. An implicit commit is performed by the system if the activation group is ending normally. An implicit rollback is performed by the system if the activation group is ending abnormally.

Note: The implicit end performed by the system for activation-group-level commitment definitions does not apply for the default activation group. This is because the default activation group is never ended when a program running within it returns. The default activation group persists for the life of the job.

Regardless of the scope for a particular commitment definition, any pending changes for a commitment definition at process end or during IPL recovery processing are always rolled back. This is true unless the process or system ended in the middle of a commit operation for that commitment definition. In that case, the commit operation is completed for the commitment definition.

Top | "Journal and Commit APIs," on page 1 | APIs by category

Code license and disclaimer information

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

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

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

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

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

Appendix. Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Programming interface information

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

Trademarks

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

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

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

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

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

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

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

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

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

Terms and conditions

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

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

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

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

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

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

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

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

Printed in USA