IBM

iSeries

Journal management

Version 5 Release 3

iSeries

Journal management

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 287.

Third Edition (October 2004)

This edition applies to version 5, release 3, modification 0 of IBM Operating System/400 (product number 5722-SSI) 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.

© Copyright International Business Machines Corporation 2002, 2004. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2002, 2004

Journal management

Journal management provides a means by which you can record the activity of objects on your system. When you use journal management, you create an object called a **journal.** The journal records the activities of the objects you specify in the form of **journal entries.** The journal writes the journal entries in another object called a **journal receiver.**

Journal management provides you with the following:

- Decreased recovery time after an abnormal end
- Powerful recovery functions
- Powerful audit functions
- The ability to replicate journal entries on a remote system

This topic provides information about how to set up, manage, and troubleshoot system-managed access-path protection (SMAPP), local journals, and remote journals on an iSeries server.

What's new for V5R3
Highlights the changes and improvements made to Journal management.

Print this topic
Print this topic to view a hardcopy of Journal management.

System-managed access-path protection
System-managed access-path protection (SMAPP) allows you to use some of the advantages of journaling without explicitly setting up journaling. Use SMAPP to decrease the time it takes to restart your system after an abnormal end.

Local journal management
Use local journal management to recover the changes to an object that have occurred since the object was last saved or provide an audit trail of changes. Use this information to set up, manage, and troubleshoot journaling on a local server. Use the Journal entry information finder to see information about journal codes and entry types.

Remote journal management
Use remote journal management to establish journals and journal receivers on a remote system that are associated with specific journals and journal receivers on a local system. Remote journal management replicates journal entries from the local system to the journals and journal receivers that are located on the remote system after they have been established.

Related information
View the manuals, IBM(R) Redbooks(TM) (in PDF format), and web sites that relate to Journal management.

Note: Read the Code example disclaimer for important legal information.

© Copyright IBM Corp. 2002, 2004

What’s new

For V5R3, there are a number of improvements and additions to journal management. The following items contain a summary of these improvements and additions.

Default for SMAPP is set to 60 minutes from 70 minutes

- How SMAPP works

Enhancements to change journaled objects without ending journaling

- Change journaling attributes of journaled objects without ending journaling

Expanded sequence numbers for your journal receivers

- Receiver size options for journals

Recovery enhancements for journaling

You can create an output file that details all the activity that occurs when you do an apply or remove journal entry operation. When you apply or remove journaled changes, a new option allows you to continue or end the operation when the system finds errors. The new default is to end the apply or remove operation for the one object that encounters the error and continue the operation for other objects. Also, the new default for applying journaled changes is to honor commitment boundaries. In previous releases, the default was to ignore commitment boundaries. Support is also added for restoring objects with partial transactions.

- Use the apply and remove journaled changes output file
- Apply journaled changes
- Actions of applying or removing journaled changes by journal code
- Example: Recover objects with partial transactions

New API for journal entries

A new API allows you to replay a database operation from one journal entry.

- Replay a database operation from a single journal entry

Information enhancements

Information enhancements include a FAQ for journaling and disk arms. Also, the journal entry information finder includes entry-specific data for journal code B integrated file system entries.

- Journaling and disk arm FAQ
- Journal entry information finder

How to see what’s new or changed

To help you see where technical changes have been made, this information uses:

- The » image to mark where new or changed information begins.
- The « image to mark where new or changed information ends.

» To find other information about what’s new or changed this release, see the Memo to Users. «

Print this topic

To view or download the PDF version, select Journal management (3.2 MB).

You can view or download these related topics:

Database programming (3.1 MB) contains the following topics:

- Setting up a database on an iSeries(TM) server.
- Using a database on an iSeries server.

Integrated file system (1.4 MB) contains the following topics:

- What is the integrated file system?
- Integrated file system concepts and terminology.
- The interfaces you can use to interact with the integrated file system.

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As...**
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

System-managed access-path protection

System-managed access-path protection (SMAPP) allows you to use some of the advantages of journaling without explicitly setting up journaling. SMAPP is a way to reduce the time for an iSeries(TM) server or independent disk pool to restart after an abnormal end. An **access path** describes the order in which records in a database file are processed. A file can have multiple access paths, if different programs need to see the records in different sequences.

When the system or an independent disk pool ends abnormally, the system must rebuild the access paths the next time you restart the system, or vary on an independent disk pool. When the system must rebuild access paths, the next restart or vary on operation takes longer to complete than if the system ended normally.

When you use SMAPP, the system protects the access paths so the system does not need to rebuild the access paths after an abnormal end. This topic introduces SMAPP, describes SMAPP concepts, and provides setup and management tasks.

SMAPP concepts
Use this information to find out why you might want to use SMAPP, how it works, and how it affects your system.

- Benefits of SMAPP
- How SMAPP works
- How the system chooses access paths to protect
- Effects of SMAPP on performance and storage

- How SMAPP handles changes in disk pool configuration
- SMAPP and access path journaling
- SMAPP and independent disk pools

Start or change SMAPP and display SMAPP status
Use this information to start or change SMAPP and to display the status of SMAPP on your server.

- Start SMAPP or change SMAPP values
- Display SMAPP status

Benefits of SMAPP

System-managed access-path protection (SMAPP) can greatly reduce the amount of time it takes to restart your system or vary on an independent disk pool, after an abnormal end. The time is reduced by protecting access paths. A protected access path can be recovered much quicker than a non-protected access path. It is an automatic function that runs without attention. SMAPP determines which access paths to protect without any intervention by the user. It adjusts to changes in the environment, such as the addition of new applications or new hardware.

SMAPP does not require any setup. You do not have to change your applications. You do not have to journal any physical files or even use journaling at all. You simply need to determine your policy for access path recovery:

- After a failure, how long you can afford to spend rebuilding access paths when you restart the system, or vary on an independent disk pool.
- How to balance access path protection with other demands on system resources.
- Whether to have different target times for recovering access paths for different disk pools.

You may need to experiment with different target recovery times for access paths to achieve the correct balance for your system. If you configure additional basic or independent disk pools, you must also evaluate your access path recovery times.

The system protects access paths by journaling the access paths to internal system journals. Therefore, SMAPP requires some additional auxiliary storage for journal receivers. However, SMAPP is designed to keep the additional disk usage to a minimum. SMAPP manages journal receivers and removes them from the system as soon as they are no longer needed.

How SMAPP works

The purpose of system-managed access-path protection (SMAPP) is to reduce the amount of time it takes to restart the system or vary on an independent disk pool, after an abnormal end.

It can take much longer than normal to restart the system when the system ends abnormally because of something like a power interruption. Also, if you are using an independent disk pool, the next vary on of the independent disk pool can take much longer than normal.

Access paths

An **access path** describes the order in which records in a database file are processed. A file can have multiple access paths, if different programs need to see the records in different sequences.

How SMAPP works with abnormal ends

When the system restarts after an abnormal end, the system rebuilds access paths that were open for updating at the time of the abnormal end. Rebuilding access paths contributes to this long restart time. Likewise, when you vary on an independent disk pool, the system rebuilds access paths that were open for updating at the time the independent disk pool ended abnormally. The system does not rebuild access paths that are specified as MAINT(*REBLD) when you create them. When protecting access paths with SMAPP, the system uses information that it has collected to bring access paths up to date, rather than rebuilding them.

You can specify the target time for rebuilding access paths after the system ends abnormally. The target time is a goal that the system does its best to achieve. The actual recovery time for access paths after a specific failure may be somewhat more or less than this target.

The target recovery time for access paths can be specified for the entire system or for individual disk pools. The system dynamically selects which access paths to protect to meet this target. It periodically estimates how long it will take to recover access paths that are open for change.

» For new systems, the system-wide recovery time for access paths is 60 minutes, which is the default. If you move from a release that does not provide the SMAPP function to a release that supports SMAPP, the system-wide recovery time for access paths is also set to 60 minutes. «

How the system chooses access paths to protect

The system periodically examines access path exposure and estimates how long it would take to rebuild all the exposed access paths. If the rebuild time exceeds your target recovery times for access paths, the system selects additional access paths for protection.

An access path is **exposed** when the access path has changed because records have been added or deleted or because a key field has changed, and those changes have not yet been written to the disk. The system periodically examines access path exposure and estimates the time required to rebuild all the exposed access paths. If the rebuild time exceeds your target recovery times for access paths, the system selects additional access paths for protection. The system can also remove access paths from protection if the estimated time for rebuilding access paths consistently falls below your target recovery times for access paths. The recover attribute of a file is not used in determining whether to protect access paths.

Some access paths are not eligible for protection by SMAPP:

- A file that specifies MAINT(*REBLD).
- An access path that is already explicitly journaled.
- An access path in the QTEMP library.
- An access path whose underlying physical files are journaled to different journals.
- Any encoded vector access path.
- » Any access path that uses an international component for unicode (ICU) sort sequence table.«
- A file journaled to a journal in standby state.

You can use the Display Recovery for Access Paths (DSPRCYAP) command to see a list of access paths that are not eligible for SMAPP.

Effects of SMAPP on performance and storage

System-managed access-path protection (SMAPP) is designed to have minimal effect to your system. Though it is minimal, SMAPP does affect your system's processor performance and auxiliary storage.

»

Processor performance

SMAPP has some affect on processor performance. The lower the target recovery time you specify for access paths, the greater this effect may be. Typically, the effect on processor performance is not very noticeable, unless the processor is nearing capacity. Another situation that may cause an increase in processor consumption is when local journals are placed in standby state and large access paths built over files journaled to the local journal are modified. The presence of standby state flags the access paths as not eligible for SMAPP protection. This may force SMAPP to protect many other small access paths in an attempt to achieve the specified target recovery time, and this can lead to performance concerns. To alleviate the processor performance impact, INCACCPTH(*ELIGIBLE) can be specified on the Change Recovery for Access Paths (CHGRCYAP) command. This will give SMAPP permission to ignore any access paths built over files journaled to journals in this state which in turn will prevent SMAPP from having to protect many other small access paths. However, this INCACCPTH option will ignore these access paths when estimating the IPL or independent Auxilary Storage Pool (ASP) vary on exposure which means that the actual IPL or independent ASP vary on duration may be longer than the estimated value.

«

Auxiliary storage

SMAPP causes increased disk activity, which increases the load on disk input/output processors. Because the disk write operations for SMAPP do not happen at the same time, they do not directly affect the response time for a specific transaction. However, the increased disk activity might affect overall response time.

Also when you use SMAPP, the system creates an internal journal and journal receiver for each disk pool on your system. The journal receivers that SMAPP uses take additional auxiliary storage. If the target recovery time for access paths for a disk pool is set to *NONE, the journal receiver has no entries. The internal journal receivers are spread across all the arms in a disk pool, up to a maximum of 100 arms.

The system manages the journal receivers automatically to minimize the affect as much as possible. It regularly discards internal journal receivers that are no longer needed for recovery and recovers the disk space. The internal journal receivers that are used by SMAPP require less auxiliary storage than the journal receivers for explicit journaling of access paths. Internal journal receivers are more condensed because they are used only for SMAPP entries.

If you have already set up journaling for a physical file, the system uses the same journal to protect any access paths that are associated with that physical file. If the system chooses to protect additional access paths, your journal receivers will grow larger more quickly. You will need to change journal receivers more often.

Tips to reduce SMAPP's affect on auxiliary storage

- When you set up SMAPP, specify target recovery times for access paths either for the entire server or for individual disk pools, but not for both. If you specify both, the system does extra work by balancing the overall target with the individual targets.
- If you also journal physical files, to deal with the increased size of your journal receivers, consider specifying to remove internal entries when you set up journaling or swap journal receivers. If you specify this, the system periodically removes internal entries from user journal receivers when it no longer needs them to recover access paths. This prevents your journal receivers from growing excessively large because of SMAPP.
- If your system cannot support dedicating any resources to SMAPP, you can specify *OFF for the system target recovery time. Before choosing this option, consider setting the recovery time to *NONE for a normal business cycle, perhaps a week. During that time, periodically display the estimated recovery time for access paths. Evaluate whether those times are acceptable or whether you need to dedicate some system resources to protecting access paths.

  If you turn SMAPP off, any disk storage that has already been used will be recovered shortly thereafter. If you set the SMAPP values to *NONE, any disk storage that has already been used will be recovered after the next time you restart your system.

Note:
If you want to change the target system recovery time to a different value after you have set it to *OFF, the system must be in a restricted state.

For more information about removing internal entries, see Receiver size options for journals. See the Performance topic for more information about system performance.

How SMAPP handles changes in disk pool configuration

When you restart the system, the system checks to see if your disk pool configuration has changed. The system may change either the size of the SMAPP receiver or the placement of the receiver based on the change to the disk units. The system considers the performance of the disk units assigned to a disk pool to determine where to place the SMAPP journal receiver.

When you restart your system, the system checks to see if your disk pool configuration has changed. The system does the following:

- If any disk units have been added or removed from an existing disk pool, the system may change either the size of the SMAPP receiver or the placement of the receiver.
- If any new disk pools are in the configuration and do not have any access path recovery times assigned for SMAPP, the system assigns a recovery time of *NONE for that disk pool. If you remove a disk pool from your configuration and later add it back, the access path for that disk pool is set to *NONE, even if that disk pool previously had a recovery time for access paths.
- If all basic user disk pools have been removed from your configuration so that you have only the system disk pool, the system access path recovery time is set to the lower of the following values:
  - The existing system access path recovery time.
  - The current access path recovery time for disk pool 1. If the current access path recovery time for disk pool 1 is *NONE, the system access path recovery time is not changed.

When you vary on an independent disk pool, the system checks to see if any disk units have been added or removed from the independent disk pool. The system may change either the size of the SMAPP receiver or the placement of the receiver based on the change to the disk units. If this is the first time the independent disk pool is varied on, then the system assigns a recovery time of *NONE for that independent disk pool.

When you add disk units to your disk configuration while your system is active, or your independent disk pool is varied on, the system does not consider those changes in making SMAPP storage decisions until the next time you restart the system, or vary on the independent disk pool. The system uses the size of the disk pool to determine the threshold size for SMAPP receivers. If you add disk units, the system does not increase the threshold size for the receivers until the next time you restart the system restart or vary on the independent disk pool. This means that the frequency of changing SMAPP receivers will not go down until you restart the system, or vary off the independent disk pool.

When you create a new user disk pool while your system is active, add all of the planned disks to the disk pool at the same time. The system uses the initial size of the new disk pool to make storage decisions for SMAPP. If you later add more disk units to the disk pool, those disk units are not considered until the next time you restart the system or vary on the independent disk pool. When you create a new user disk pool, the access path recovery time for that disk pool is set to *NONE. You can use the EDTRCYAP command to set a target recovery time for the new disk pool, if desired.

For more information about disk pools and how to manage them, see Manage disk units in disk pools.

SMAPP and access path journaling

In addition to using system-managed access path protection (SMAPP), you can choose to journal some access paths yourself by using the Start Journaling Access Path (STRJRNAP) command. This is called

explicit journaling. To journal an access path explicitly, you must first journal all the underlying physical files. SMAPP does not require that the underlying physical files be journaled.

The reason for choosing to journal an access path explicitly is that you consider the access path (and the underlying files) absolutely critical. You want to make sure that the files are available as soon as possible when the system is started after an abnormal end.

Under SMAPP, the system looks at all access paths to determine how it can meet the specified target times for recovering access paths. It may not choose to protect an access path that you consider critical.

When the system determines how to meet the target times for recovering access paths, it considers only access paths that are not explicitly journaled.

How SMAPP is different from explicitly journaling access paths:

- SMAPP does not require that underlying physical files be journaled.
- SMAPP determines which access paths to protect based strictly on the target recovery times for all access paths. You might choose to journal an access path explicitly because of your requirements for the availability of a specific file.
- SMAPP continually evaluates which access paths to protect and responds to changes in your server environment.
- SMAPP does not require any user intervention to manage its internal journals and journal receivers.
- SMAPP uses less disk space for journal receivers because they are detached and deleted regularly.

For more information about when to journal access paths, see Reasons to journal access paths.

SMAPP and independent disk pools

You can use SMAPP to protect access paths for independent disk pools. When you use SMAPP to protect access paths in independent disk pools, you can specify the recovery time individually for each independent disk pool. This improves the performance when you vary on your independent disk pool after an abnormal vary off.

The recovery time that you specify moves with the independent disk pool if you switch it between systems. Therefore if you are switching an independent disk pool between systems, you only need to specify the recovery time once.

The only time the specified recovery time is not moved is when the system you are moving the independent disk pool to has its system recovery time specified as *OFF. In this case the independent disk pool's recovery time is set to *NONE when you vary on the independent disk pool.

Start SMAPP or change SMAPP values

Use the Edit Recovery Access Path (EDTRCYAP) display to start or change values for system-managed access-path protection (SMAPP).

If you use basic or independent disk pools to separate objects that have different recovery and availability requirements, you might also want to specify different recovery times for access paths in those disk pools.

For example, if you have a large history file that changes infrequently, you can put the file in a separate disk pool and set the access path recovery time for that disk pool to *NONE. Or, if you have an independent disk pool, and you want the recovery time to move with the disk pool when it is switched to another server, you can specify a specific time for that disk pool.

To start SMAPP or change SMAPP values, proceed as follows:

1. On the display, specify one of the following values in the **System access path recovery time** field:
   - *SYSDFT
   - *NONE
   - *MIN
   - *OFF
   - » A specific value between 1 and 1440 minutes.«
2. At the **Include access paths** field select one of the following:
   - *ALL
   - *ELIGIBLE
3. If you are starting or changing SMAPP for disk pools, change the **Target** field for individual disk pools.

To change the access path recovery time from *OFF to another value, your server must be in a restricted state.

You can also use the Change Recovery for Access Paths (CHGRCYAP) command to change the target recovery times without using the Edit Recovery Access Path display.

The system performance monitor also provides information about access path recovery times. Work Management on the V5R1 Supplemental Manuals Web site and Performance Tools for iSeries(TM) provide more information about monitoring performance and about what SMAPP information is available through the tools.

Display SMAPP status

You can use the Edit Recovery Access Path display to display the values for system-managed access-paths (SMAPP) are set for the following:

- The entire server.
- Basic and independent disk pools.
- Access paths not eligible for protection.
- Protected access paths.

Use the top part of the display to see the values for the entire server. Use the bottom part of the display to see the values for individual disk pools on the system. If you do not have basic or independent disk pools that are active, the bottom part of the display says `No user ASP configured or information not available`.

Estimated time for recovery

To see the number of minutes the system estimates it will need to recover most of the access paths, look at the **Estimated recovery time for access paths** field. The time is an estimated maximum, based on most circumstances. It assumes that the system is recovering access paths on a dedicated server (during a restart) and that all eligible access paths are being recovered or rebuilt. It does not include time to rebuild access paths that must be rebuilt for one of the following reasons:

- The access path is damaged.
- The access path was marked as `not valid` during a previous abnormal end and was not successfully rebuilt.
- One of the following commands marked the access path as `not valid` and was running when the system failed:
  - Copy File (CPYF), if the system chose to rebuild the access path for efficiency.

- Reorganize Physical File Member (RGZPFM)
- Restore Object (RSTOBJ)

If you have basic or independent disk pools, the estimated recovery time for access paths for the entire server (**System access path recovery time** field) might not equal the total estimated recovery time for the disk pools (**Access Path Recovery Time-Estimated (Minutes)**). When you restart the system or vary on an independent disk pool, the system overlaps processing when recovering access paths to reduce the total time it requires.

Disk space used

The **Disk Storage Used** field on the display shows the disk space that SMAPP uses only for internal system journals and journal receivers. It does not include any additional space in user-managed journal receivers for protecting access paths whose underlying physical files are already journaled.

Access paths not eligible

You can display all access paths that are not eligible for protection. To view access paths that are not eligible for protection, press F13. Access paths that are not eligible for access protection are as follows:

- Access paths built over physical files which are journaled to separate journals.
- Access paths built over a physical file which is journaled to a journal whose state is currently standby.

Protected access paths

You can also display up to 500 protected access paths by pressing F14. The system displays the access paths with the highest estimated recovery time first.

Use can also use the Display Recovery for Access Paths (DSPRCYAP) command to display or print the estimated recovery times and disk usage.

Local journal management

Use local journal management to recover the changes to an object that have occurred since the object was last saved, as an audit trail, or to help replicate an object. Setting up journaling locally is a prerequisite for other iSeries(TM) functions such as Remote journal management and Commitment control.

The Local journal management topic provides concept, planning, setup, management, and recovery information for journaling objects on a local iSeries server.

Journal management concepts
Explains how journal management works, why to use it, and how it affects your system.

Plan for journal management
Provides you with the information you need to ensure you have enough disk space, to plan what objects to journal, and to plan which journaling options to use.

Set up journaling
Provides instructions to set up journals and journal receivers.

Start and end journaling
Provides instructions to start journaling after you create journals and receivers. Also provides instructions for ending journaling.

Manage journals
Provides tasks to manage your journaling environment.

Scenario: Journal management
Provides the steps that a fictitious company, JKL Toy company, takes as it implements journal management on its iSeries server.

Recovery operations for journal management
Provides tasks that show you how to use journaling to recover data on your iSeries server.

Journal entry information
Provides information and tasks for working with journal entries.

Note: Read the Code example disclaimer for important legal information.

Journal management concepts

Journal management enables you to recover the changes to an object that have occurred since the object was last saved. You can also use journal management to provide an audit trail or to help replicate an object. You use a journal to define what objects you want to protect with journal management. The system keeps a record of changes you make to objects that are journaled and of other events that occur on the system.

This topic provides information about how journals work, information about journal entries, and how journals affect system performance:

- Benefits of journal management
- How journal management works
- Journal entries
- Journal management and system performance
- Journals with the save-while-active function

Benefits of journal management

The primary benefit of journal management is that it enables you to recover the changes to an object that have occurred since the object was last saved. This ability is especially useful if you have an unscheduled outage such as a power failure.

In addition to powerful recovery functions, journal management also has the following benefits:

- Journal management enhances system security. You can create an audit trail of activity that occurs for objects.
- Journal management allows you to generate user defined journal entries to record activity, even for objects that do not allow journaling.
- Journal management provides quicker recovery of access paths if your system ends abnormally.
- Journal management provides quicker recovery when restoring from save-while-active media.
- » Journal management provides the means to recover an object that was saved with partial transactions.

Save your server while it is active has instructions for saving an object with partial transactions. Example: Recover objects with partial transactions has instructions for recovering objects with partial transactions.

«

How journal management works

When you use journal management you create an object called a **journal**. You use a journal to define which objects you want to protect. You can have more than one journal on your system. A journal can define protection for more than one object.

You can journal the objects that are listed below:

- Database physical files
- Access paths
- Data areas
- Data queues
- Integrated file system objects (stream files, directories, and symbolic links).

Journal entries

The system keeps a record of changes you make to objects that are journaled and of other events that occur on the system. These records are called journal entries. You can also write journal entries for events that you want to record, or for objects other than the object that you want to protect with journaling.

For example, some journal entries identify activity for a specific database record such as add, update, or delete. (If the updated object image after the update is the same as the image before the update, then journal entries are not deposited for that update.) Also journal entries identify activity such as a save, open, or close operation for an object. Journal entries can also identify other events that occur, such as security-relevant events on the system or changes made by dynamic performance tuning. Journal entry information describes all the possible journal entry types and their contents.

Each journal entry can include additional control information that identifies the source of the activity, including the user, job, program, time, and date. The entries that the system deposits for a journaled object reflect the changes made to that journaled object. For example, the entries for changes to database records can include the entire image of the database record, not just the changed information.

Journal receivers

The system writes entries to an object called a **journal receiver**. The system sends entries for all the objects associated with a particular journal to the same journal receiver.

You can attach journal receivers to a journal by using iSeries(TM) Navigator or the Create Journal (CRTJRN) and Change Journal (CHGJRN) commands. The system adds journal entries to the attached receiver. Journal receivers that are no longer attached to a journal and are still known to the system are **associated** with that journal. Use the Work with Journal Attributes (WRKJRNA) command to see a list of receivers associated with a journal.

The system adds an entry to the attached journal receiver when an event occurs to a journaled object. The system numbers each entry sequentially. For example, it adds an entry when you change a record in a journaled database file member. Journal entries contain information that identifies:

- Type of change
- Record that has been changed
- Change that has been made to the record
- Information about the change (such as the job being run and the time of the change)

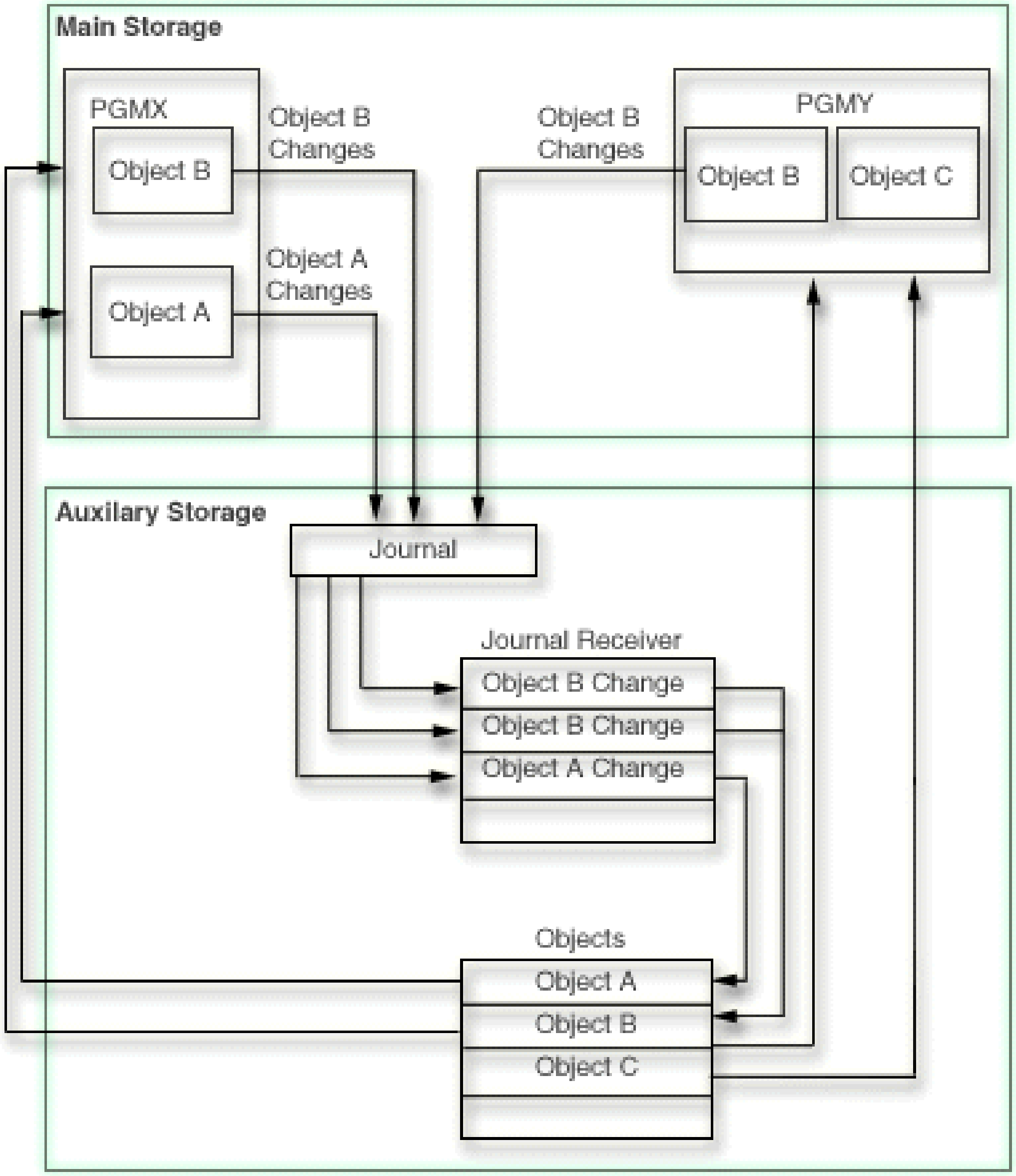
When you are journaling objects, changes to the objects are added to the journal receiver. The system does not journal data that you retrieved but did not change. If the logical file record format of a database file does not contain all the fields that are in the dependent physical file record format, the journal entry still contains all the fields of the physical file record format. In addition, if you are journaling access paths, entries for those access paths are added to the journal. If the updated physical file image after the update is the same as the image before the update, and if the file has no variable length fields, then journal entries are not deposited for that update. If the updated data area image after the update is the same as the image before the update, then journal entries are not deposited for that update. If the attribute that was requested to be changed was already that value, then journal entries are not deposited for that change.

Summary of the journaling process

The following figure shows a summary of journal processing. Objects A and B are journaled; object C is not. Programs PGMX and PGMY use object B. When you make a change to object A or B, the following occurs:

- The change is added to the attached journal receiver.
- The journal receiver is written to auxiliary storage.
- The changes are written to the main storage copy of the object.

Object C changes are written directly to the main storage copy of the object because it is not being journaled. Only the entries added to the journal receiver are written immediately to auxiliary storage. Changes against the object may stay in main storage until the object is closed.



You can also take advantage of the remote journal function. The remote journal function allows you to associate a journal on a remote system with a journal on a local system. Journal entries on the local system are replicated to the remote journal receiver.

Journal entries

When you use journal management, the system keeps a record of changes that you make to objects that are journaled and of other events that occur on the system. These records are called journal entries. You can use journal entries to help recover objects or analyze changes that were made to the objects.

Every journal entry is stored internally in a compressed format. The operating system must convert journal entries to an external form before you can see them. You cannot change or access the journal entries directly. Not even the security officer can remove or change journal entries in a journal receiver. You can use these journal entries to help you recover your objects or analyze changes that were made to the objects.

Contents of a journal entry

Journal entries contain the following information:

- Information that identifies the type of change.
- Information that identifies the data that was changed.
- The after-image of the data.
- Optionally, the before-image of the data (this is a separate entry in the journal).
- Information that identifies the job, the user, and the time of change.
- The journal identifier of the object.
- Information that indicates if the entry-specific data is minimized.

The system also places entries in the journal that are not for a particular journaled object. These entries contain information about the operation of the system and the control of the journal receivers.

Journal identifier

When you start journaling an object, the system assigns a unique **journal identifier** (JID) to that object. The system uses the JID to associate the journal entry with the corresponding journaled object. For more detailed information about journal identifiers, see Why you must save objects after you start journaling

Journal entry numbering

Each journal entry is sequentially numbered without any missing numbers until you reset the sequence number with the Change Journal (CHGJRN) command or iSeries$^{(TM)}$ Navigator. However, when you display journal entries, sequence numbers can be missing because the system uses some entries only internally. For audit purposes, you can display most of these internal entries with the INCHIDENT option on the Display Journal (DSPJRN) command. However, the DSPJRN command will not display journal entries that are from a release that is newer than the release on your system. This situation can occur from one of the following:

- You restore the journal receiver to a previous release.
- You use remote journaling to replicate a journal receiver to a system with a previous release.

When the system exceeds the largest sequence number, a message is sent to the system operator identifying the condition and requesting action. No other journal entries can be added to the journal until the journal receivers are changed and the sequence number is reset.

Fixed-length and variable-length portions

A journal entry that is converted for displaying or processing contains a fixed-length prefix portion that is followed by a variable-length portion. The variable-length portion contains entry-specific data and, in some cases, null-values indicator data. The format of the converted entry depends on the command that you use and the format that you specify. The entry-specific data varies by entry type. The Send Journal Entry (SNDJRNE) command or the QJOSJRNE API specifies the entry-specific data for user-created journal entries.

Methods for working with journal entries

The following list contain links to CL commands and APIs for working with journal entries:

- Apply Journaled Changes (APYJRNCHG) command
  »
- Apply Journaled Changes Extend (APYJRNCHGX) command
  «
- Compare Journal Images (CMPJRNIMG) command
- Delete Pointer Handle (QjoDeletePointerHandle) API
- Display Journal (DSPJRN) command
- Get Path Name of Object from Its File ID (Qp0lGetPathFromFileID()) API
- Receive Journal Entry (RCVJRNE) command
- Retrieve Journal Entry (RTVJRNE) command
- Retrieve Journal Entries (QjoRetrieveJournalEntries) API
- Remove Journaled Changes (RMVJRNCHG) command
- Send Journal Entry (SNDJRNE) command
- Send Journal Entry (QJOSJRNE) API
- Send Journal Entry (QJOSJRNE) API »
- Replay Database Operation (QDBREPLAY) API «

Journal entry information has details on all possible journal entries and the information associated with them. »

Journal management and system performance

Journal management prevents transactions from being lost if your system ends abnormally or has to be recovered. Journal management writes changes to journaled objects immediately to the journal receiver in auxiliary storage. Journaling increases the disk activity on your system and can have a noticeable affect on system performance.

Journaling also increases the overhead associated with opening objects and closing objects, as the number of objects you are journaling increases, the general performance of the system can be slower. The time it takes to perform an IPL on your system or vary on of an independent auxiliary storage pool (ASP) can also increase, particularly if your system or independent ASP ends abnormally.

The system takes measures to minimize the performance effect of using journaling features. For example, the system packages before-images, after-images, and any access path changes for a record in a single write operation to auxiliary storage. Therefore, journaling access paths and before-images, in addition to after-images, usually does not cause additional performance overhead. However, they do add to the auxiliary storage requirements for journaling.

The system also spreads journal receivers across multiple disk units to improve performance. If you do not specify a maximum receiver-size option, then the system can place the journal receiver on up to ten disk units in a disk pool. If you specify a maximum receiver-size option, and a matching sufficiently large journal size threshold then the system can place the journal receiver on up to 100 disk units in a disk pool.

You can take measures to minimize the effect of journaling on your system performance:

- Consider using journal caching. Journal caching is a separately chargeable feature that causes the system to write journal entries to memory in large groups. When there are several journal entries in memory then the system writes journal entries from memory to disk. If the application performs a large number of changes, this can result in fewer synchronous disk writes resulting in improved performance. However, when you use journal caching, a few of the most recent updates to your journaled objects may be lost on an abnormal IPL or independent ASP vary on.
- Before using journal standby state, consider the potential System Managed Access Path Protection (SMAPP) impacts of making that choice and consider specifying INCACCPTH(*ELIGIBLE) on the Change Recovery for Access Paths (CHGRCYAP) command. For further discussion of these impacts see the article on standby state. «
- Do not set the force-write ratio (FRCRATIO) parameter for physical files that you are journaling. You can let the system manage when to write records for the physical file to disk because the journal receiver has a force-write ratio of 1.
- » Isolate journal receivers in a disk pool that is not the system disk pool, if the separate disk pool has at least two disk arms. This reduces contention when accessing the disks. If the separate disk pool does not have at least two disk arms, there is no performance benefit. If the I/O processor (IOP) for your disk units have at least 25 MB of write cache, your performance can be better if the disk units in the disk pool are protected through device parity protection rather than mirrored.«
- Consider using record blocking when a program processes a journaled file sequentially (SEQONLY(*YES)). When you add or insert records to the file, the records are not written to the journal receiver until the block is filled. You can specify record blocking with the Override with Database File (OVRDBF) command or in some high-level language programs. If you use the OVRDBF command, do the following:
  - Set the SEQONLY parameter to (*YES).
  - Use a large enough value for the NBRRCDS parameter to make the buffer approach the optimal size of 128KB.
- Consider minimizing the fixed-length portion of the journal entry using RCVSIZOPT(*MINFIXLEN) for the journal. When you specify this option, all of the data that is selectable by the FIXLENDTA parameter is not deposited. Therefore, that information does not have to be retrieved, benefiting journal performance.
- Consider omitting information from the journal entry you do not need using the OMTJRNE parameter. When you specify the OMTJRNE parameter for database physical files you will not deposit the file open and close entries which saves processing as well as disk storage space. Similarly, if you specify the OMTJRNE parameter for directories and stream files, the object open, close, and force entries are not deposited.
- Ensure you have enough write cache for your I/O processor (IOP).

For more information about journaling and disk arm usage, see the Journaling and disk arm FAQ. See the Performance topic for more information about system performance. The Manage disk units in disk pools topic has information about disk pools, disk units, and disk protection. The Striving for Optimal Journal Performance on DB2 Universal Database(TM) for iSeries(TM) redbook has detailed information about improving journal performance.

Journal management with the save-while-active function

Journaling can help you with recovery if you use the save-while-active function in your backup strategy. If you plan to save an application without ending it for checkpoint processing, consider journaling all of the objects associated with the application. After the save operation is complete, save all of the journal receivers for the objects you are saving.

If you need to do a recovery, you can restore objects from the save-while-active media. Then you can apply journal changes to an application boundary.

» You also can use the save-while-active function to save an object with partial transactions—before the transactions reach a commit boundary. When you restore an object with partial transactions, you cannot use it without additional actions. Journaling enables you to apply or remove changes to an object with partial transactions to restore it to a usable state.

Using the save-while-active function to save your journaled objects can help you recover your objects more quickly when you need to apply or remove journaled changes specifying FROMENT(*LASTSAVE) or FROMENTLRG(*LASTSAVE). When you use the save-while-active function to save your journaled objects, the system saves and then restores information that indicates which starting journal sequence number is needed for the apply or remove operation. When this information is available for all objects to which you are applying or removing journaled changes, the system does not need to scan the journal receivers to determine this starting point. Scanning journal receiver data to find the starting points can be time consuming.

Also, using the save-while-active function when saving your objects allows you to restore a version of your object which was not from the last save and to still specify FROMENT(*LASTSAVE) or FROMENTLRG(*LASTSAVE) on the apply or remove command and successfully apply or remove changes.

See Save your server while it is active for more information about the save-while-active function. See Example: Recover objects with partial transactions for information about restoring objects with partial transactions to a usable state. The Commitment control topic has more detailed information about transactions. «

Plan for journal management

Before you start to journal an object, you must make decisions that will determine how you will create journals and receivers, what objects to journal and how to journal those objects. These decisions include:

- Whether to use iSeries(TM) Navigator to set up your journaling environment.
- What objects to protect with journaling.
- Whether to journal other objects that the system does not journal.
- Whether to combine journaling with the save-while-active function.
- How many journals you need and which objects must be assigned to each journal.
- Whether to journal after-images only or both before-images and after-images.
- Whether your application programs must write journal entries to assist with recovery.
- What type of disk pool in which to store your journal receiver
- Whether to use the remote journal function to replicate the journal entries and receivers to one or more additional systems. »
- Whether to omit the optional open, close, or force entries for your objects. «

You also need to make operational decisions about journal management:

- How often must journal receivers be changed and saved?
- How often must you save journaled objects?
- How must journals and journal receivers be secured?

Finally, you need to balance the benefits of journaling with the affect it may have on your system performance and auxiliary storage requirements.

Use the following information to help you make these decisions:

- iSeries Navigator versus character-based interface for journaling objects
- Plan which objects to journal
- Plan for journal use of auxiliary storage

- Plan setup for journal receivers
- Plan setup for journals

Remote journal management has information about remote journaling.

iSeries(TM) Navigator versus the character-based interface for journaling objects

There are two environments that you can use for journal management: iSeries Navigator and the character-based interface. iSeries Navigator provides a graphical interface for journaling that is easy to use and does not require the use of control language (CL) commands. The character-based interface requires the use of CL commands or APIs, but has more functionality than iSeries Navigator.

The following is a list of journaling functions that are only available with the character-based interface:
- Journal access paths.
- » Specify a maximum receiver-size option.«
- Specify that objects allow journal entries to have minimized entry-specific data.
- Specify the data that is included in the fixed-length portion of the journal entries.
- Specify the time to delay the next attempt to automatically attach or delete a new journal receiver with system journal-receiver management.
- Specify journal caching.
- Specify journal standby state.
- Compare journal entries.
- Apply journaled changes.
- Remove journaled changes.
- Display journal entries.
- » Look at the journal to display all object types which are journaled to that journal.«
- » Change the journaling attributes for a journaled object without ending journaling.«

Other journaling differences between iSeries Navigator and the character-based interface are as follows:
- With iSeries Navigator, you create the journal and journal receiver together. With the character-based interface, you create the journal receiver first.
- With iSeries Navigator you set the permissions for the journal and receiver after they are created. With the character-based interface you can set permissions (authority) at creation time.

» Decide which of the two interfaces to use before you set up journal management, since the character-based interface creates journal receivers and journals separately, and iSeries Navigator creates journals and receivers together. However, if you decide to use a function that iSeries Navigator does not support after you start journaling, you can do so with the character-based interface, even if you used iSeries Navigator to set up journaling. «

Plan which objects to journal

When you plan which objects to journal, consider the following:
- What types of objects you can journal
- What makes an object a good candidate for journaling
- What rules for journaling apply to those objects
- Whether or not to send journal entries for objects the system does not journal

Types of objects that are eligible for journaling

You can journal the following object types:
- Database physical files
- Access paths

- Data areas
- Data queues
- Integrated file system objects (stream files, directories, and symbolic links)

General characteristics that make objects good candidates for journaling

- An object with a high volume of transactions between save operations is a good candidate for journaling.
- An object that is difficult to reconstruct the changes made to it, such as an object that receives many changes without physical documentation. For example, an object used for telephone order entry is more difficult to reconstruct than an object used for orders that arrive in the mail on order forms.
- An object that contains critical information. For example, if you restore an object back to the last save operation, and the delay from reconstructing changes to that object has a negative effect on your operation: that object is a good candidate for journaling.
- Objects that relate to other objects on the system. Although the information in a particular object may not change often, that object may be critical to other, more dynamic objects on the system. For example, many files may depend on a customer master file. If you are reconstructing orders, the customer master file must include new customers or changes in credit limits.
- Objects that require that all the actions on it be replicated.
- An object, that, after a crash, has a requirement to be recovered to a consistent state and have a journal entry show what actions completed.
- An object that can cause a negative consequence to your operation if a crash damages that object while the system is in the process of updating it.
- An object for which you want to have an audit trail of changes.

Considerations for journaling database physical files

- If you journal one file that participates in a referential constraint, you must journal all the related files. Referential constraints are not enforced when you apply or remove journaled changes, but the referential integrity of those constraints is verified.
- If you journal all related files, the process for applying and removing journaled changes keeps the relationships between your database files valid. If you do not journal all related files, your referential constraint may show a status of **check pending** after you apply or remove journaled changes. For some types of referential constraints, the system requires that you journal all of the related files.
- For a file that has a trigger program, if the trigger program only performs processing on object types which can be journaled and applied, you must journal all of the objects processed by the trigger program. If the trigger programs do additional work that must be reconstructed during a recovery, consider using the API support for sending journal entries.
- In general, database source files must not be journaled. If you use the Start Source Entry Utility (STRSEU) command to update a member, every record in that member is considered changed and every record is journaled to the journal. However, if changes to a source file are critical, you can journal the file in the same manner as data files.

Considerations for journaling integrated file system objects

- When you start journaling on a symbolic link, the link is not followed. Therefore if you want to protect the actual object with journaling, you have to journal the actual object separately.
- If you want to automatically protect all objects which are created in a directory which itself is journaled, consider the use and impacts of the inherit journaling attribute that you can associate with a journaled directory.
- Do you want to protect the structure of the directory tree, or just the data stored in stream files within that directory structure? If you just want to protect the data stored in stream files, then for performances reasons, it may be best to only journal the stream files themselves instead of journaling changes to each directory in the directory tree. You must consider this question when you use the subtree and inherit journaling attributes options on the start journaling interfaces.

- You cannot journal objects on a user-defined file system (UDFS) independent disk pool. If you want to journal objects in a UDFS, you must use a library capable independent disk pool. Journal management and independent disk pools has more information about journaling and independent disk pools.

System objects

It is recommended that you do not journal changes to IBM(R)-supplied objects. The system sometimes creates and manages these objects differently than user-created objects. The system does not assure the recovery of these files even though all recovery activity normally succeeds.

Journal entries for objects the system does not journal

Some applications depend on information in objects that the server does not journal. For example, an application programming interface (API) might use a user space to pass data between two jobs.

You can use the Send Journal Entry (SNDJRNE) command or the Send Journal Entry (QJOSJRNE) API to write journal entries for these resources. See Send your own journal entries for instructions. If you need to do recovery, you can use a program to retrieve these journal entries and make sure these application objects are synchronized with the objects you are journaling.

If you are using commitment control, you can use APIs to register these objects as committable resources.

Before images and access paths

- Reasons to journal access paths has detailed information about whether or not to journal access paths.
- Reasons to journal before-images discusses whether or not to journal before-images

Reasons to journal access paths: If you journal access paths, the system can use the journal entries to recover access paths instead of rebuilding them completely.

When your server ends abnormally, perhaps because of a power interruption, the next IPL can take much longer than a normal IPL. Rebuilding access paths contributes to this long IPL time. When you perform an IPL after an abnormal end, the system rebuilds access paths that were exposed, except those access paths that are specified as MAINT(*REBLD) when you create the file. An access path is exposed if changes have been made to it that have not been written to the disk.

If you journal access paths, the system can use the journal entries to recover access paths instead of rebuilding them completely. This reduces the time it takes to IPL after the system ends abnormally. Access path journaling is strictly for the purpose of server recovery during an IPL. You do not use access path journal entries when you are applying journal changes to recover a file.

If certain access paths and their underlying files are critical to your operation, you want to ensure that these files are available as soon as possible after the system ends abnormally. You can choose to journal these access paths. This is called **explicit access path journaling.** .

Explicit access path journaling differs system-managed access-path protection (SMAPP) in that with SMAPP you cannot control which access paths the system chooses to protect. Therefore, if the system does not protect the access path that you consider critical to meet your target recovery times, you must explicitly journal that access path.

If you choose to journal an access path, remember the following:

- You can journal an access path for a physical file only if the physical file has a keyed access path or an index created by a referential constraint.
- Before you start journaling an access path, you must journal all the underlying physical files to the same journal.
- You can journal only access paths that are defined as MAINT(*IMMED) or MAINT(*DLY).

- You cannot journal encoded vector access paths.
- » You cannot journal an access path with an international components for unicode (ICU) sort sequence table. You can journal access paths with other sort sequence tables.«

The System-managed access-path protection topic has detailed information about SMAPP.

Reasons to journal before-images: When you journal an object, the system always writes an after-image for every change that is made. You can request that the system write before-image journal entries for database files and data areas. All other object types only journal after-images. This significantly increases the auxiliary storage requirements for journaling.

However, you can choose to journal before-images for these reasons:

- Before-images are required for a backout recovery, where you remove journal changes with the Remove Journaled Changes (RMVJRNCHG) command rather than applying journal changes to a restored copy of an object. Backout recovery is often complex, particularly if multiple users and programs are accessing the same object. It is most commonly used when new applications or programs are being tested.
- For database physical files, before-images are required to use the Compare Journal Images (CMPJRNIMG) command. This command highlights the differences between the before-images and after-images. It is sometimes used to audit changes to a database file.
- For database physical files, if you want a copy of the record that is deleted to be part of the deleted record journal entry information, you must specify before-images.
- Commitment control requires before-images for the system to roll back uncommitted changes. When you open a database file under commitment control, the system automatically journals both before-images and after-images while the commitment definition is active. If you normally journal only after-images, the system writes before-images only for the changes made under commitment control. If the system initiates the journaling of before-images, you cannot use them to remove journaled changes. Commitment control does not support integrated file system objects, data areas, or data queues.
- Access path journaling also requires before-images for the system to use for IPL recovery. When you journal access paths, or the system journals an access path to provide system-managed access-path protection, the system will automatically journal both before and after-images. If you normally journal only after-images, the system also writes before-images if you are journaling the access path.

» You can select before-images on an object-by-object basis. You specify whether you want after-images or both when you start journaling for a database file or a data area. After you start journaling a database file or a data area, you can use the Change Journaled Object (CHGJRNOBJ) command to change whether you are journaling before-images. «

Plan for journal use of auxiliary storage

If you are journaling an object, journal management writes a copy of every object change to the journal receiver. It writes additional entries for object level activity, such as opening and closing the object, adding a member, or changing an object attribute. If you have a busy system and journal many objects, your journal receivers can quickly become very large.

The maximum size for a single journal receiver varies. It depends on how the system allocates the journal receiver across multiple disk arms. The maximum size ranges from approximately 1.9 GB to 1.0 TB depending on what value you specified for the associated journal's receiver size option.

» To avoid possible problems with a journal receiver exceeding the maximum size allowed on the system, specify a threshold for the receiver of no more than 900 000 000 KB if you specified a journal receiver maximum-size option for the associated journal. Otherwise, specify a threshold of no more than 1 441 000 KB. «

The following topics provide more information about how journal management affects auxiliary storage:

- Functions that increase the journal receiver size
- Methods to estimate the size of a journal receiver
- Journal receiver calculator
- Methods to reduce the storage that journal receivers use
- Determine the type of disk pool in which to place journal receivers
- Journals and independent disk pools

Frequently asked questions about journaling and disk arm usage:

Journaling affects the disk arms that store the journal receiver. How the journal receiver affects the disk arm depend on several factors:

- The threshold setting you are using for your journal receiver
- Whether you are using a maximum receiver-size option
- The way in which the system writes journal entries to disk

The following are frequently asked questions about journaling and disk arm usage:

How many arms in my disk unit will journaling use?
Which journal parameters and settings affect the number of the disk arms the journal receiver uses?
Why is the system not using the new arms I added to my disk pool?
Why are some disk arms used by journal receivers noticeably busier than the others and what can I do to spread out the usage?

How many arms in my disk unit will journaling use?

How many disk arms the journal receiver uses depends on your threshold value and whether you use a maximum receiver-size option. When you create a journal receiver, the system spreads the journal receiver on up to 10 disk arms. If you use a maximum receiver-size option, the system spreads the journal receiver on up to 100 disk arms. Some rules that the system uses when determining the number of disk arms are as follows:

- The system cannot use more disk arms than are available in your disk pool.
- The system will not use fewer than 10 disk arms if that many arms are available.

You can use the following formula to determine how many disk arms you will use:

```
Number of disk arms = Journal Threshold setting / 64 MB
```

The release of 0S/400 you have also determines the number of disk arms journaling uses. If you use a release before V5R2, and specify to remove internal entries, the system sets aside a third of the arms for internal entries (specifically transient index entries). Also before V5R2 these arms are dedicated to the internal entries. For example, if you have 35 disk arms on your system, and have a threshold large enough to include 20 arms, the system uses an additional ten arms for the internal entries. The system leaves only five arms for that specific journal receiver.

Starting with V5R2, the system uses a different scheme for disk arm use by internal entries. The internal entries are spread across all of the disk arms on the system. This is potentially all 35 arms in the previous example.

For more information about disk arm use and journaling see Striving for Optimal Journal Performance on DB2 Universal Database(TM) for iSeries(TM)

Back to questions

Which journal parameters and settings affect the number of the disk arms the journal receiver uses? The threshold for the journal receiver and whether you use a maximum receiver-size option have the largest effect on how many disk arms the journal receiver uses. If you have a system which is before V5R2, removing internal entries also affects the number of disk arms that are used.

Back to questions

Why is the system not using the new disk arms I added to my disk pool?

There can be a several reasons. First, to use the newly added disk arms, you must perform a change journal operation to attach a new journal receiver. Also, the system does not necessarily use all of the disk arms in a disk pool. If you are not using a maximum receiver-size option, the most disk arms the system will spread the receiver over is ten. The number of disk arms the receiver uses also depends on the threshold you use for your journal receiver. If you use a maximum receiver-size option and increase your threshold, it is more likely that your new disk arm will be used.

If you use system-managed access-path protection (SMAPP), the system generates internal journal entries to protect the access paths for database files. If you have not upgraded to at least V5R2, setting your journal receiver to remove internal entries is an issue if you are not producing these internal entries. Before V5R2, removing internal entries can steal disk arms from the normal journal entries. For example, if you have six disk arms in the disk pool housing your journal receiver and remove internal entries, two arms are dedicated to the internal entries and four arms are used for your regular journal entries. If you do not produce any internal entries, those two arms remained idle. For V5R2 and later, this is not an issue.

For more information about disk arm use and journaling see Striving for Optimal Journal Performance on DB2 Universal Database for iSeries

Back to questions

Why are some disk arms used by journal receivers noticeably busier than the others and what can I do to spread out the usage?
The journal receivers probably use some disk arms more than other because of the way journal management writes journal entries to disk. When the system produces journal entries, journal management stores the journal entries in memory. When it is ready, journal management sends the journal entries to a disk arm in one group. When the next group of journal entries are ready, journal management sends the entries to the next disk arm. Journal management continues in this sequential manner until all of the disk arms it uses have received a group of journal entries. The cycle then repeats.

You can spread out the usage by increasing your threshold and using a maximum receiver-size option.

For more information about disk arm use and journaling see Striving for Optimal Journal Performance on DB2 Universal Database for iSeries

Back to questions

«

Functions that increase the journal receiver size: Some optional functions available with journal management can significantly increase auxiliary storage requirements.

You can select to journal both before-images and after-images. The system uses more storage if you select both before-images and after-images, although storage use is not necessarily doubled. If you journal access paths, the before-images and after-images are written to the journal receiver when a database file is updated. Only after-images are written when a database file is added (write operation) or deleted. Neither the before-image nor after-image is deposited into the journal if the after-image is exactly the same as the before-image.

Using Fixed-length options for journal entries can also increase auxiliary storage requirements. The additional storage that fixed-length options use is similar to the extra space that is used by journaling both before-images after-images.

The system requires additional space to journal access paths. The space required depends on the following items:

- How many access paths are journaled.
- How often you change the access paths. When you update a record in a database file, you cause an access path journal entry only if you update a field included in the access path.
- The method used to update access paths. More journal entries are written if you update access paths randomly than if you update them in ascending or descending sequence. Doing a mass change to an access path field, such as a date change, causes the fewest journal entries.

If you are using system-managed access-path protection and you journal database files, the system uses the same journal receiver to protect access paths for that file. This also increases the size of your journal receivers.

The information in Methods to estimate a journal receiver will help you predict your requirements for auxiliary storage.

Methods to estimate the size of a journal receiver: You can use the methods below to estimate the effect a journal receiver will have on auxiliary storage.

The actual auxiliary storage used will be somewhat larger because the system writes additional entries for such actions as opening and closing objects unless you to omit open and close journal entries when start journaling for database physical files or integrated file system objects.

Method 1- Journal receiver calculator

Use Journal receiver calculator. The Journal receiver calculator provides an easy way for you to estimate the size of your journal receiver without setting up journaling.

The calculator assumes the following:

- You are journaling after-images only.
- You are using a single journal receiver for an entire day’s transactions.
- You are journaling database physical files only. It does not include estimates for access path journaling, integrated file system objects, data areas, data queues, or user-created entries.
- You are not minimizing entry-specific data for the files.

Method 2 - Running a test

Another method for estimating the size of the journal receiver is to run a test. This method is more accurate because it includes all journal entries. Additionally, this method will work for any object type which can be journaled, not just database physical files unlike method one. To use this method, you must either have journaling set up already or you must set it up.

If you are already using journaling, skip steps 1 and 2 below. Instead, issue a Display Journal Receiver Attributes (DSPJRNRCVA) command before the time period so you can compare sizes from the beginning of the period to the end.

This method assumes that the same receiver is used during the whole test. If there is a change journal to attach a new journal receiver during the test, you must include the sizes of all the receivers.

1. Set up journaling by creating the receiver and journal.
2. Start journaling for all the objects that you plan to journal.
3. Choose a time period (1 hour) with typical transaction rates.
4. After one hour, use the Display Journal Receiver Attributes (DSPJRNRCVA) command to display the size of the receiver.
5. Multiple the size by the number of hours that your system is active in a day.

Estimate the size of the journal receiver manually: Use this procedure to estimate the size of your journal receiver.

This procedure assumes the following:

- You are journaling after-images only.
- You are using a single journal receiver for an entire day's transactions.
- You are journaling database physical files only. It does not include estimates for access path journaling, integrated file system objects, data areas, data queues, or user-created entries.
- You are not using the MINENTDTA parameter to minimize entry-specific data for the files.

Follow the steps below to estimate the size of a journal receiver:

1. Determine the average record length for all the files that you plan to journal. If the record lengths vary significantly and the information is available, use a weighted average based on the relative number of transactions per file.
2. If you are not minimizing the fixed-length portion of the journal entry (not specifying RCVSIZOPT(*MINFIXLEN) on the CRTJRN command), you can specify the data that is included in the fixed-length portion (FIXLENDTA) of the journal entries. Find the sum of the bytes for the options you are using. Select the options from the following list:

   *JOB = 26 bytes
   *USR = 10 bytes
   *PGM = 10 bytes
   *PGMLIB = 22 bytes
   *SYSSEQ = 8 bytes
   *RMTADR = 20 bytes
   *THD = 8 bytes
   *LUW = 27 bytes
   *XID = 140 bytes
3. Estimate the number of transactions for a day.
4. The system-created portion of a journal entry is approximately 50 bytes. (It varies by the type of journal entry.)
5. Estimate the number bytes of auxiliary storage needed for one day's transactions by using the following formula:

   ```
   Total bytes needed = (a+b+50)*c
   ```

   where:

   a = the average record length of files (step 1)
   b = sum of values selected for FIXLENDTA (step 2)
   c = number of transactions for a day (step 3)

For example:

1. The average record length for journaled files is 115 bytes.
2. *JOB, *USR, and *PGM options of FIXLENDTA are selected. Their sum is 46 bytes.

3. The number of journaled transactions per day is 10 000.
4. The total bytes needed to journal after-images for a day is:

   ```
   (115+46+50) * 10 000 = 2 110 000
   ```

Methods to reduce the storage that journal receivers use: You can reduce the size of journal entries by methods such as journaling after-images only, or specifying certain journaling options including the Fixed Length Data (FIXLENDTA) option on the Create Journal (CRTJRN) and Change Journal (CHGJRN) commands. Methods to reduce the storage needed for journaling are as follows:

»

Journal after-images only
Unless you are using commitment control, after-images are sufficient for your recovery needs. When you start journaling, the default is to journal after-images only. You can use the Change Journaled Object (CHGJRNOBJ) command to stop journaling before-images without ending journaling for that object.

Omit the journal entries for open, close or force operations to journaled objects
You can omit these journal entries with the OMTJRNE parameter on the Start Journal Physical file (STRJRNPF) or Start Journal (STRJRN) command. For database files (tables), you can select **Exclude open and close entries** when you start journaling with iSeries(TM) Navigator. For integrated file system objects, ensure that **Include open, close, and synchronization entries** is not selected when you start journaling with iSeries Navigator. You can also use the CHGJRNOBJ command to start omitting these journal entries for objects that you are currently journaling. «

Omitting these journal entries can have a noticeable effect on both space and performance if an application opens, closes, or forces objects frequently. However, if you omit the journal entries for opening and closing objects, you cannot perform the following tasks:

- Use open and close boundaries when applying or removing journal changes (the TOJOBO and TOJOBC parameters).
- Audit which users open particular objects.

Swap journal receivers, save them, and free storage more frequently
Frequently saving and freeing storage for journal receivers help reduce the auxiliary storage that the receivers use. However, moving journal receivers off-line increases your recovery time because receivers have to be restored before journal changes can be applied.

Specify receiver size options that can decrease journal receiver size
Specifying the following receiver size options can help reduce journal receivers size:

- Remove internal entries. This causes the system to periodically remove internal entries that it no longer needs, such as access path entries.
- Minimize the fixed-length portion the journal entry. This causes the system to no longer deposit all of the data selectable by the FIXLENDTA parameter in the journal entry, thus reducing the size of the entries. However, if you require this journal entry information for audit or other uses, you cannot use this storage saving technique. Additionally, it reduces the options available as selection criteria used on the following commands and API:
  - Display Journal (DSPJRN)
  - Receiver Journal Entry (RCVJRNE)
  - Retrieve Journal Entry (RTVJRNE)
  - Compare Journal Images (CMPJRNIMG)
  - Apply Journaled Changes (APYJRNCHG)
  - » Apply Journaled Changes Extend (APYJRNCHGX)«
  - Remove Journaled Changes (RMVJRNCHG)

- Retrieve Journaled Entries (QjoRetrieveJournalEntries) API

Minimized entry-specific data for journals
Minimizing entry-specific data allows the system to write data to the journal entries in a minimized format.

Select the fixed-length options for data carefully
Fixed-length options can quickly increase the size of your journal receiver. The journal receiver calculator can help you determine the effect of fixed-length options on your auxiliary storage.

If you are journaling a physical file, specify `SHARE(*YES)` for the file.
You can do this using the Create Physical File (CRTPF) command or the Change Physical File (CHGPF) command. The system writes a single open and close entry regardless of how often the shared open data path (ODP) is opened or closed within a routing step.

Determine the type of disk pool in which to place journal receivers: You can use disk pools (auxiliary storage pool) to control which objects are allocated to which groups of disk units. If you are journaling many active objects to the same journal, the journal receiver can become a performance bottleneck. One way to minimize the performance impact of journaling is to put the journal receiver in a separate disk pool. This also provides additional protection because your objects are on different disk units from the journal receiver, which contains a copy of changes to the objects.

iSeries$^{(TM)}$ servers have several types of disk pools:

System disk pool
The system disk pool contains the operating system. It can also contain user libraries and objects. The system disk pool is always disk pool number 1.

Basic disk pool
Basic disk pools are disk pool numbers 2 through 32. A basic disk pool can be a library or a non library disk pool. The differences are as follows:

- A library disk pool contains one or more user libraries or user-defined file systems. It does not contain the operating system. This is the current recommended method of configuring user disk pools.
- A non library disk pool contains no user libraries or user-defined file systems. It may contain journals, journal receivers, and save files. If you place a journal receiver in a non library basic disk pool, the journal must be in either the system disk pool or the same non library disk pool. The journaled objects must be in the system disk pool.

Independent disk pool
Independent disk pool are disk pools 33 through 255. If you use independent disk pools, you can only put journals and journal receivers on independent disk pools that are library capable. If you are going to place the journal receiver in a switchable independent disk pool, the journal receiver, the journal, and journaled object must be in the same disk pool group (though they do not have to be in the same disk pool).

When disk pools were first introduced, they were called auxiliary storage pools (ASPs). Only non library user ASPs were available. Many systems still have this type of ASP. However, recovery steps are more complex for non library user ASPs. Therefore, for systems implementing journaling for the first time, library disk pools are recommended.

Journal management and independent disk pools has more specific information about using journaling with independent disk pools. Manage disk units in disk pools has specific information about disk pools. The Independent disk pools topic has detailed information about setting up independent disk pools.

Journal management and independent disk pools: Independent disk pools are disk pools 33 through 255. Independent disk pools can be user-defined file system (UDFS) independent disk pools or library-capable independent disk pools.

UDFS and library-capable independent disk pools

UDFS independent disk pools are independent disk pools that only have a user-defined file system. UDFS independent disk pools cannot store journals and receivers. In contrast to UDFS disk pools, library-capable independent disk pools have libraries and are capable of storing journals and receivers. If you plan to journal objects on an independent disk pool, you must use a library-capable independent disk pool.

Note: A library-capable independent disk pool can have integrated file system objects. You can also journal integrated file system objects on a library-capable independent disk pool.

You cannot journal objects on a UDFS independent disk pool.

Switchable and dedicated independent disk pools

Independent disk pools can also be switchable or dedicated. Dedicated independent disk pools are used on only one system. Switchable independent disk pools can be switched between systems. If they are library-capable, you can journal objects on either switchable or dedicated independent disk pools.

Disk pool groups

You can group switchable independent disk pools into disk pool groups. Disk pool groups consist of one primary disk pool and one or more secondary disk pools. If you are going to journal an object in a disk pool group, the object and the journal must be in the same disk pool. The journal receiver can be in a different disk pool, but must be in the same disk pool group as the journal and journaled object.

Rules for journaling objects on independent disk pools

Use the following rules when journaling objects on independent disk pools:

- The disk pool must be available on the system on which you are working.
- The disk pool must be a library-capable disk pool. You cannot journal an object on a UDFS independent disk pool.
- In a disk pool group, the journaled object and the journal must be in the same disk pool.
- In a disk pool group, the journal receiver can be in a different disk pool, but must be in the same disk pool group.

Manage disk units in disk pools has information about managing disk pools. The Independent disk pools topic has information about setting up and managing independent disk pools.

Plan setup for journal receivers

The following topics provide information to plan configuration for journal receivers. They provide information about each option that you can select for journal receivers.

- Disk pool assignment for journal receivers
- Library assignment for journal receivers
- Naming conventions for journal receivers
- Threshold (disk space) for journal receivers
- Security for journal receivers

Disk pool assignment for journal receivers: Placing journal receivers in a different disk pool from the journaled objects may prevent performance bottlenecks.

Before you place the journal receiver in a library basic disk pool, you must first create the library for the journal receiver in the disk pool.

You can only place a journal receiver in an independent disk pool if the independent disk pool is library capable. If you are placing the journal receiver in a switchable independent disk pool, you must place it in the same disk pool group as the journal and the object you are journaling. Manage disk units in disk pools has more information about disk pools. The Independent disk pools topic has detailed information about independent disk pools.

If you are creating the journal receiver with the Create Journal Receiver (CRTJRNRCV) command, you can use the ASP parameter to allocate storage space for the journal receiver in a different disk pool (ASP) than the library to which you assigned the journal receiver. Do this only if the disk pool is a basic nonlibrary disk pool.

Library assignment for journal receivers: When you create a journal receiver, you specify a qualified name that includes the library for the receiver. The library must exist before you create the journal receiver.

You can assign a library from either the **New Journal** dialog in iSeries(TM) Navigator or with the Create Journal Receiver (CRTJRNRCV) command.

Naming conventions for journal receivers: When you create a journal receiver, either with iSeries(TM) Navigator or the Create Journal Receiver (CRTJRNRCV) command, you assign a name to the journal receiver. When you use iSeries Navigator or the Change Journal (CHGJRN) command to detach the current journal receiver and create and attach a new receiver, you can assign a name or have the system generate one. If you use system journal-receiver management, the system generates the name when it detaches a receiver and creates and attaches a new one.

If you plan to have more than one journal on your system, use a naming convention that links each journal with its associated receiver.

To simplify recovery and avoid confusion, make each journal receiver name unique for your entire system, not unique within a library. If you have two journal receivers with the same name in different libraries and they both become damaged, the reclaim storage operation renames both journal receivers when they are placed in the QRCL library. When you use the Rename Object (MOVOBJ) command for a journal or journal receiver in the QRCL library, you can change the name of the library back to the original name. You cannot change the name of the journal or the journal receiver.

» When you detach the receiver from the journal and attach a new one, you can have the system generate the name for the new receiver by incrementing the previous receiver name. If you use system change-journal management by specifying MNGRCV(*SYSTEM) for the journal, the system also generates a new receiver name when it changes journal receivers. The default for the Create Journal (CRTJRN) command is to use system change-journal management. «

The following table shows the rules the system uses to generate a new receiver name. It applies these rules in the sequence shown in the table.

Current name	System action	Example
Last 4 characters are numeric.	Adds 1	DSTR0001 to DSTR0002
Last character is not numeric.	Truncates the name to 6 characters, if necessary. Adds 0001	DSTRCVR to DSTRCV0001

Current name	System action	Example
Last character is numeric. Last non-numeric character is in position 5 or less.	Adds 1	DSTR01 to DSTR02
Last character is numeric. Last non-numeric character is in position 6 or higher.	Truncates to 6 characters, if necessary. Adds 0001.	DSTRCVR01 to DSTRCV0001

If you restore a journal to your system, the system creates a new journal receiver and attaches it to the journal. The system generates a name for the new journal receiver based on the name of the journal receiver that was attached when the journal was saved. The following table shows the rules the system uses to generate a new receiver name when you restore a journal:

Current name	System action	Example
Last 4 or more characters are numeric.	Adds 1 to the leftmost digit of the numeric portion.	DSTR0001 to DSTR1001
Last character is not numeric.	Truncates to six characters, if necessary. Adds 1000.	DSTRCVR to DSTRCV1000.
Ending numeric portion is less than 4 digits.	Pads the left portion of the numeric portion with zeroes to create a 4-digit suffix. Adds 1 to the leftmost digit.	DSTRCV01 to DSTRCV1001.

If the name generated by the system is the same as the name of a journal receiver already on the system, the system adds 1 to the name until it creates a name that is not a duplicate. For example, assume a journal receiver named RCV1 was attached when the journal was saved. When the journal is restored, the system attempts to create a new journal receiver named RCV1001. If that name already exists, the system tries the name RCV1002.

The following table shows examples of how the system generates new receiver names:

Last journal receiver known to the system[1]	Created by change journal[2]	Created by restoring journal
A	A0001	A1000
ABCDEF	ABCDEF0001	ABCDEF1000
ABCDEFG	ABCDEF0001 [3]	ABCDEF1000 [3]
ABCDEF1234	ABCDEF1235	ABCDEF2234
A0001	A0002	A1001
A1	A2	A1001
A9	A10	A1009
ABCDEF7	ABCDEF0001 [3]	ABCDEF1007 [3]
ABCDEF9999	Error [4]	ABCDEF0999
A1B15	A1B16	A1B1015

<table>
<tr><th>Last journal receiver known to the system[1]</th><th>Created by change journal[2]</th><th>Created by restoring journal</th></tr>
<tr><td colspan="3"><b>Notes:</b>
[1] If the journal exists on the system, the last journal receiver known to the system is the journal receiver that is currently attached. If the journal does not exist, the last journal receiver known to the system is the journal receiver that was attached when the journal was saved.
[2] Either when a user issues the CHGJRN command with JRNRCV(*GEN) or when the journal is changed by system change-journal management.
[3] The last character of the current name is dropped because it exceeds 6 characters.
[4] If the journal is set up as MNGRCV(*SYSTEM), the receiver name wraps around to 0's (ABCDEF0000). If the journal is set up as MNGRCV(*USER), an error occurs because adding 1 to 9999 causes an overflow condition.</td></tr>
</table>

Threshold (disk space) for journal receivers: » When you create a journal receiver with iSeries(TM) Navigator or the Create Journal Receiver (CRTJRNRCV) command, you specify a disk space threshold that indicates when you want the system to warn you or take action. When the receiver reaches that threshold, the system takes the action specified in the manage receiver (MNGRCV) parameter for the journal. The default storage threshold is 1 500 000 KB. «

In specifying a storage threshold, you need to balance the amount of space that you have available with the additional system resources that are used to change journal receivers frequently. Consider these options:

Base the size on your available auxiliary storage:

1. Calculate the amount of auxiliary storage space that you have available in the user ASP for the journal receiver.
2. Assign a receiver threshold that is 75 to 80 percent of that space.

Base the size on how often you want to change journal receivers:

1. Use the one of the methods described in Methods to estimate the size of a journal receiver to calculate how large your receiver can be for a day. If you are just journaling database physical files, you can use the Journal receiver calculator to estimate the size of your journal receiver.
2. Determine how many times a day you will detach and save the journal receiver.
3. Divide the result of step 1 by the result of step 2. This is your receiver threshold.

» Do not make the journal receiver size too small, or the system will spend too much resource changing journal receivers or sending threshold messages. To avoid possible problems with a journal receiver exceeding the maximum size allowed on the system, specify a threshold for the receiver of no more than 900 000 000 KB if you specify a maximum receiver-size option for the associated journal. Otherwise, specify a threshold of no more than 1 441 000 KB. «

Manual versus system journal-receiver management. discusses options for managing your journal receivers.

Security for journal receivers: If a journal receiver has confidential data, someone with authority to that journal receiver could possibly display that confidential data.

When you create a journal receiver, you specify the authority that all users on the system have to access it (public authority). The default authority for the Create Journal Receiver (CRTJRNRCV) command and iSeries(TM) Navigator is *LIBCRTAUT, which means the system uses the value of the create authority (CRTAUT) parameter for the journal receiver's library.

When you create a journal receiver with iSeries Navigator, you set permissions (authority) after you create the journal receiver.

Journal receivers contain copies of changes from all objects being journaled. Someone with access to the journal receiver could display confidential data. The authority to a journal receiver must be as strict as the authority for the most confidential object that is being journaled.

You do not need any authority to the journal or to the journal receiver to use an object that is journaled. Authority to the journal receiver is checked only when using commands that operate directly on the receiver. The authority you set for the journal receiver has no effect on the people using the journaled objects. iSeries Security Reference has more information about the authority required to access objects and perform commands that use journals and journal receivers.

Plan setup for journals

The following topics provide information to plan configuration for journals. They provide information about each option that you can select for journal.

- Disk pool assignment for journals
- Library assignment for journals
- Naming conventions for journals
- Journal and journal receiver association
- Journal message queue
- Manual versus system journal receiver management
- Automatic deletion of journal receivers
- Receiver size options for journals
- Minimized entry-specific data for journal entries
- Fixed-length options for journal entries
- Journal cache
- Object assignment to journals

Disk pool assignment for journals: If you want to place the journal in a library basic disk pool, you must first create the library for the journal in the disk pool. If you use a library basic disk pool, the journal and all the objects you are journaling to it must be in the same library basic disk pool.

You can only place a journal in an independent disk pool if the independent disk pool is library capable. If you are placing the journal in a switchable independent disk pool, you must place it in the same disk pool group as the journal receiver associated with the journal. Manage disk units in disk pools has more information about disk pools. The Independent disk pools topic has information about independent disk pools.

If you want to place the journal in a non library basic disk pool, you must first create the library for the journal in the system disk pool. If the journal is in a non library basic disk pool, all the objects being journaled to it must be in the system disk pool.

If you are creating the journal with the Create Journal (CRTJRN) command, you can use the ASP parameter to allocate storage space for the journal in a different disk pool (ASP) than the library to which you assigned the journal. Do this only if the disk pool is a basic nonlibrary disk pool.

Library assignment for journals: When you create a journal, you specify a qualified name that includes the library for the journal. The library must exist before you create the journal.

You can assign a library from either iSeries$^{(TM)}$ Navigator or with the Create Journal (CRTJRN) command.

Naming conventions for journals: When you create a journal with iSeries$^{(TM)}$ Navigator or the Create Journal (CRTJRN) command, you assign a name to it. If you plan to have more than one journal on your system, use a naming convention that links each journal with its associated receiver.

To simplify recovery and avoid confusion, make each journal name unique for your entire system, not unique within a library. If you have two journals with the same name in different libraries and they both become damaged, the reclaim storage operation renames both journals when they are placed in the QRCL library. When you use the MOVOBJ command for a journal in the QRCL library, you can change the name of the library back to the original library name. You cannot change the name of the journal itself. In this case, you would not be able to recover your journal from QRCL since its name has been changed.

Naming conventions to ensure restore sequence

Name the libraries for the journals, objects, and journal receivers to ensure that the objects are restored in the correct order. A naming convention will ensure that the system automatically starts journaling after a restore operation. To ensure that journaling is automatically started again, the journals must be restored before the objects being journaled. (If the journals and associated objects are in the same library, then the system automatically restores the objects in the correct order.)

If you start the name of the library for the journal with a special character, such as #, $, or @, the system will restore the library for the journal before the library for the objects. This is because in normal sort sequence, special characters appear before alphabetic characters.

When the journals and associated objects are in different libraries, you must ensure that the objects are restored in the correct order.

Since independent file system objects do not exist in a library, your restore processing must ensure the objects are restored in the correct order. That is, you must restore your libraries which contain the journals before restoring the independent file system objects that were journaled to that journal.

Journal and journal receiver association: When you create a journal, you must specify the name of the journal receiver to be attached to it. If you are using the Create Journal (CRTJRN) command to create the journal, the journal receiver must exist before you can create the journal. The receiver that you attach may not have been previously attached to a different journal or have been interrupted while being attached to any journal. You can specify up to two journal receivers, but the system ignores the second receiver.

With iSeries$^{(TM)}$ Navigator, you simply create the journal. When you create the journal, iSeries Navigator creates the journal receiver in the library you specify in the **New Journal** dialog.

Journal message queue: When you create or change a journal, you can specify where the system sends messages that are associated with the journal. In addition, you can create a program to monitor this message queue and handle any messages associated with the journal. The system also sends messages that are related to the remote journal function to this message queue.

A common use for this message queue is to handle threshold messages. When you create a journal receiver, you can specify a storage threshold. If you choose to change journal receivers yourself, you can specify where the system sends messages when the journal receiver exceeds its storage threshold. You can create a special message queue for this purpose and create a program to monitor the message queue for message CPF7099. When the message is received, the program can, for example, detach the receiver and save it.

If you specify that the system manages the journal receiver, the system does not send a threshold message. Instead, when the system automatically changes the journal receiver, it sends message CPF7020, which indicates that it successfully detached the journal receiver.

There are other messages which are sent to this journal message queue related to processing for the Delete Receiver (DLTRCV) option of the Create Journal (CRTJRN) command. See Delete journal receivers for more information.

For iSeries$^{(TM)}$ Navigator, you select the message queue in the **Advanced Journal Attributes** or **Journal Properties** dialogs. For the character-based interface, you can select the message queue with the Create Journal (CRTJRN) or Change Journal (CHGJRN) command

See Threshold (disk space) for journal receivers for information about storage threshold. See Manual versus system journal-receiver management for methods to specify journal receiver management.

Manual versus system journal-receiver management: When you create a journal with iSeries$^{(TM)}$ Navigator or the Create Journal (CRTJRN) command, you can choose one of two options to manage journal receivers:

- User journal-receiver management
- System journal-receiver management

» The default for the CRTJRN command is to have the system manage the journal receivers. «

User journal-receiver management

If you specify user journal receiver management, you are responsible for changing the journal receiver when it approaches its storage threshold. If you choose this option, you can have the system send a message to a message queue when the journal receiver approaches its storage threshold.

System journal-receiver management

If you use system journal-receiver management, you can avoid having to do some journal management chores. However, if you are journaling for recovery purposes, you need to ensure that you save all journal receivers that have not been saved, not just the currently attached receiver. Also, if you are journaling for recovery purposes, be sure to specify that the system does not automatically delete receivers when no longer needed. Automatic deletion of journal receivers describes this option.

If you use system journal-receiver management, you must ensure that your environment is suitable and that you regularly check the QSYSOPR message queue and the message queues assigned to your journals.

If the system cannot complete the change journal operation because it cannot obtain the necessary locks, it retries every 10 minutes (or as specified by the MNGRCVDLY parameter). It sends messages (CPI70E5) to the journal's message queue and to the QSYSOPR message queue. If this occurs, you may want to determine why the operation cannot be performed and either correct the condition or swap the journal receiver your self with iSeries Navigator or the CHGJRN command.

If the system cannot complete the change journal operation for any reason other than lock conflicts, it temporarily discontinues system journal-receiver management for that journal and sends a message (CPI70E3) to the message queue assigned to the journal or to the QSYSOPR message queue. This might occur because a journal receiver already exists with the name that it would generate. Look at the messages in the QHST job log to determine the problem. After you correct the problem, perform a swap journal operation to do the following:

- Create a new journal receiver
- Detach the current receiver and attach a new journal receiver

- The system then resumes system journal-receiver management.

System journal-receiver management when you restart the system

» When you restart the system or vary on an independent disk pool, the system performs a CHGJRN command to change the journal receiver and reset the journal sequence number.

Note: If you are using maximum receiver-size option *MAXOPT3, the sequence number is not reset when you restart the system or vary on an independent disk pool.

Also, if the journal is attached while a maximum receiver-size option is specified, the system attempts to perform a CHGJRN command to reset the sequence number when the following is true:

- When the sequence number exceeds 9 900 000 000 if RCVSIZOPT(*MAXOPT1) or RCVSIZOPT (*MAXOPT2) is in effect for the journal.
- When the sequence number exceeds 18 446 644 000 000 000 000 if RCVSIZOPT(*MAXOPT3) is in effect for the journal.

For all other journal receivers, the system attempts this CHGJRN when the sequence number exceeds 2 147 000 000. «

The system does not reset the journal sequence number when you restart the system or vary on an independent disk pool if the entries in the receiver may be needed for commitment control recovery.

Delay automatic journal change

If you use the CRTJRN or CHGJRN command, you can use the Manage Receiver Delay Time (MNGRCVDLY) parameter. When you use system journal-receiver management for a journal, if the system cannot allocate an object needed to attach a new journal receiver to the journal, it will wait the length of time that you specify in the MNGRCVDLY parameter before its next attempt to attach the new journal receiver. If you do not specify this parameter, the system will wait ten minutes, which is the default.

» The following topics have information related to management of journal receivers:

- Automatic deletion of journal receivers
- Threshold (disk space) for journal receivers
- Swap journal receivers
- Receiver size options for journals«

Automatic deletion of journal receivers: If you choose system journal receiver management, you can also have the system delete journal receivers that are no longer needed for recovery. You can only specify this if you are using system journal receiver management. The system can only evaluate whether a receiver is needed for its own recovery functions, such as recovering access paths or rolling back committed changes. It cannot determine whether a receiver is needed to apply or remove journaled changes.

» Attention: Use automatic deletion of journal receivers with care if you use save-while-active operations to save objects before they reach a commitment boundary. Ensure that you save the journal receivers before the system deletes them. If an object is saved before it reaches a commitment boundary it can have partial transactions. To avoid data loss you must have access to the journal receivers that were attached during the save-while-active operation when you restore the objects with partial transactions. «

The system will automatically delete journal receivers if you do one of the following:

- Specify **Delete receivers when no longer needed** in the iSeries(TM) Navigator Advanced Journal Attributes or Journal Properties dialog.
- Specify DLTRCV (*YES) in the Create Journal (CRTJRN) or Change journal (CHGJRN) commands.

However, even if you select one of the previous items, the system cannot delete the journal receiver if any of the following conditions is true:

- An exit program that is registered for the Delete Journal Receiver exit point (QIBM_QJO_DLT_JRNRCV) indicates that the receiver is not eligible for deletion.
- A journal has remote journals associated with it, and one or more of the associated remote journals does not have a full copy of this receiver.
- The system could not get the appropriate locks that are required to complete the operation.
- The exit program registration facility was not available to determine if any exit programs were registered.

If you use system delete-receiver support, you must ensure that your environment is suitable. You must also regularly check the QSYSOPR message queue and the message queues that are assigned to your journals.

- If the system cannot complete the DLTJRNRCV command for any of the above reasons, it retries every 10 minutes (or the value you specify on the DLTRCVDLY parameter). It sends a CPI70E6 message to the journal's message queue, and to QSYSOPR message queue. If this occurs, you might want to determine why the operation cannot be performed and either correct the condition or run the DLTJRNRCV command.
- If the system cannot complete the command for any other reason, it sends a CPI70E1 message to the message queue that is assigned to the journal. If you have not specifically assigned a message queue to the journal, the message will be sent to the QSYSOPR message queue. Look at the messages in QHST to determine the problem. After you correct the problem, use the DLTJRNRCV command on the specific journal receiver.

Do not select to have the detached journal receiver deleted if you might need it for recovery or if you want to save it before it is deleted. The system does not save the journal receiver before deleting it. The system does not issue the warning message (CPA7025) that it sends if a user tries to delete a receiver that has not been saved.

Examples of when you might specify automatic journal deletion include:

- You are journaling only because it is required to use commitment control
- You are journaling for explicit access-path protection
- You are replicating the journal receiver to another system through the remote journal function, and that system is providing the backup copy of the journal receiver.

Delay the next attempt to delete a journal receiver

If you are using the CRTJRN or CHGJRN command, you can use the Delete Receiver Delay Time (DLTRCVDLY) parameter. The system waits the time you specify (in minutes) with the DLTRCVDLY parameter before its next attempt to delete a journal receiver that is associated with the journal when one of the following is true:

- The system cannot allocate a needed object.
- You are using an exit program, and the exit program votes no.
- You are using remote journaling and the receiver has not been replicated to all the remote journals.

If you do not specify this parameter, the system waits ten minutes, which is the default.

»

Save your server while it is active has instructions for saving an object with transactions in a partial state. Example: Recover objects with partial transactions has instructions for recovering objects with partial transactions. «

Receiver size options for journals: A journal receiver holds journal entries that you might use for recovery and entries that the system might use for recovery. For example, you might use record level entries, such as database record changes, and file level entries, such as the entry for opening or closing a file. Also, the system writes entries that you never see or use, such as entries for explicitly journaled access paths, for SMAPP, or for commitment control.

When you create a journal with the Create Journal (CRTJRN) command, the Change Journal (CHGJRN) command, or iSeries(TM) Navigator, you can specify options that will limit the data that gets deposited into these journal entries, or increases the maximum allowable size for the journal receiver. These options are as follows:

- The RCVSIZOPT parameter of the CRTJRN command
- The RCVSIZOPT parameter of the CHGJRN command
- The **Advanced Journal Attributes** dialog box of iSeries Navigator
- The **Journal Properties** dialog box of iSeries Navigator

The following subtopics explain the benefits of some of the values for receiver size options.

Remove internal entries

When you specify to remove internal entries the system periodically removes internal journal entries from the attached journal receiver when it no longer needs them for recovery purposes. Removing internal entries may have a very slight impact on system performance, because the system has to manage these internal entries separately and periodically remove them.

To remove internal entries specify the RCVSIZOPT(*RMVINTENT) parameter. The iSeries Navigator equivalent to the RCVSIZOPT(*RMVINTENT) parameter is **Remove internal entries** in the **Advanced Journal Attributes** or **Journal Properties** dialog.

Specifying to remove internal entries has these benefits:

- It reduces the impact that SMAPP may have on journal receivers for user-created journals.
- It reduces the size of journal receivers that are on the system.
- It reduces the amount of time and media required to save journal receivers, because unnecessary entries are not saved.
- It reduces the time that it takes to apply journal entries, because the system does not have to evaluate unnecessary entries.
- It reduces the communications overhead if the remote journal function is being used because unnecessary entries are not sent.

Minimize fixed-length portion of entries

Minimizing the fixed-length portion of entries has the following effects:

- All information selectable by the FIXLENDTA parameter is not deposited in the entries.
- Minimizing the fixed-length portion of entries reduces auxiliary storage space and some CPU time as well.
- When you view journal entries with this information removed, the displayed value is *OMITTED, blanks, or zeros, depending on the type of data.

- To determine if a journal receiver was attached to a journal while minimizing the fixed-length portion of entries, use the Display Journal Receiver Attributes DSPJRNRCVA command display.
- Do not use minimize the fixed-length portion of entries if you require an audit trail.
- Minimizing the fixed-length portion of entries limits the selection criteria you can use on the following:
  - Apply Journaled Changes (APYJRNCHG) command
    »
  - Apply Journaled Changes Extend (APYJRNCHGX) command
    «
  - Compare Journal Images (CMPJRNIMG) command
  - Display Journal (DSPJRN) command
  - Receive Journal Entry (RCVJRNE) command
  - Remove Journaled Changes (RMVJRNCHG) command
  - Retrieve Journal Entry (RTVJRNE) command
  - Retrieve Journal Entries (QjoRetrieveJournalEntries) API
- Minimizing the fixed-length portion of entries reduces the communications overhead if the remote journal function is being used because unnecessary data is not sent.

To minimize the fixed-length portion of entries specify RCVSIZOPT(*MINFIXLEN). The iSeries Navigator equivalent to RCVSIZOPT(*MINFIXLEN) is **Minimize fixed portion of entries** in the **Advanced Journal Attributes** or **Journal Properties** dialog.

If you are using minimizing the fixed-length portion of entries, you cannot use the FIXLENDTA parameter. See Fixed-length options for journal entries for more information about the FIXLENDTA parameter.

Maximum receiver-size options

» Use the following options to specify the maximum allowable size for your journal receivers and to specify the largest allowable sequence numbers for journal entries. There is no iSeries Navigator equivalent to the following options. «

RCVSIZOPT(*MAXOPT1)
Use RCVSIZOPT(*MAXOPT1) to set the maximum size of a journal receiver attached to your journal to 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 15 761 440 bytes. RCVSIZOPT(*MAXOPT1) is the default.

RCVSIZOPT(*MAXOPT2)
Use RCVSIZOPT(*MAXOPT2) to set the maximum size of a journal receiver attached to your journal to approximately one terabyte (1 099 511 627 776 bytes) and a maximum sequence number of 9 999 999 999. However, with RCVSIZOPT(*MAXOPT2), the system can deposit a journal entry as large as 4 000 000 000 bytes. You cannot save or restore these journal receivers to any releases before V5R1M0. Nor can you replicate them to any remote journals on any systems at a release before V5R1M0.

»

RCVSIZOPT(*MAXOPT3)
Use RCVSIZOPT(*MAXOPT3) to set the maximum size of a journal receiver attached to your journal to approximately one terabyte (1 099 511 627 776 bytes). In addition, with RCVSIZOPT(*MAXOPT3) the journal receiver can have a maximum sequence number of 18 446 744 073 709 551 600. With RCVSIZOPT(*MAXOPT3), the system can deposit a journal entry

as large as 4 000 000 000 bytes. You cannot save or restore these journal receivers to any releases before V5R3M0. Nor can you replicate them to any remote journals on any systems at a release before V5R3M0.

If you use RCVSIZOPT(*MAXOPT3) you must use the FROMENTLRG and TOENTLRG parameters to specify a journal entry sequence number larger than 9 999 999 999 when you perform the following commands:

- APYJRNCHG
- APYJRNCHGX
- CMPJRNIMG
- DSPJRN
- RCVJRNE
- RMVJRNCHG
- RTVJRNE

«

Minimized entry-specific data for journal entries: On the Create Journal (CRTJRN) and Change Journal (CHGJRN)commands, you can specify to make minimized journal entries. This will decrease the size of your journal entries. When you specify the Minimized Entry Specific Data (MINENTDTA) parameter for an object type, the entry-specific data for the entries of those object types can be minimized. You can minimize journal entries for database physical files and data areas.

The system only minimizes entries if the minimized entry is smaller in size than a complete journal entry deposit would be. Therefore, even if you specify this option, not all entries that are deposited will be minimized. The Display Journal (DSPJRN) command, Receiver Journal Entry (RCVJRNE) command, Retrieve Journal Entry (RTVJRNE) command, and QjoRetrieveJournalEntries API return data that indicates whether the entry is actually minimized or not.

You cannot save or restore a journal receiver with minimized journal entries to any release prior to V5R1M0, nor can they be replicated to any remote journal on a system at a release prior to V5R1M0.

See Journal code finder to see which entries are eligible for minimization. See Considerations for entries which contain minimized entry-specific data for more information and considerations when using these entries.

Fixed-length options for journal entries: You can use the Fixed Length Data (FIXLENDTA) parameter of Create Journal (CRTJRN) and Change Journal (CHGJRN) commands to audit security related activity for journaled objects on your system. With the FIXLENDTA parameter, you can elect to include security related information in the fixed-length portion of the journal entries. You cannot use the FIXLENDTA parameter and Minimize fixed-length portion of entries at the same time.

Fixed-length options

With the FIXLENDTA parameter, you can specify that the following data is included in the journal entries that are deposited into the attached journal receiver:

Job name
Use the *JOB value to specify the job name.

User profile name
Use the *USR value to specify the effective user profile name.

Program name
Use the *PGM value to specify the program name.

Program library name
Use the *PGMLIB value to specify the program library name and the auxiliary storage pool device name that contains the program library.

System sequence number
Use the *SYSSEQ value to specify the system sequence number. The system sequence number gives a relative sequence to all journal entries in all journal receivers on the system.

Remote address
Use the *RMTADR value to specify the remote address, the address family and the remote port.

Thread identifier
Use the *THD value to specify the thread identifier. The thread identifier helps distinguish between multiple threads running in the same job.

Logical unit of work identifier
Use the *LUW value to specify the logical unit of work identifier. The logical unit of work identifies work related to specific commit cycles.

Transaction identifier
Use the *XID value to specify the transaction identifier. The transaction identifier identifies transactions related to specific commit cycles.

Journal cache: Journal caching is separately chargeable feature with which you can specify that the system cache journal entries in main storage, before writing them to disk. Journal caching is option 42 of the OS/400$^{(R)}$ operating system.

After you have you purchased journal caching, you can specify it with the JRNCACHE parameter on the Create Journal (CRTJRN) or Change Journal (CHGJRN) commands.

Journal caching provides significant performance improvement for batch applications which perform large numbers of changes to the data portion of the journaled objects. The actions that show a performance improvement if journal caching is enabled are as follows:

- Changes to database files from add, update, or delete operations
- Changes to data areas from uses of the change data area command or API
- Changes to data queues from uses of the send data queue API or the receive data queue API
- Changes to integrated file system objects from various write and fclear operations on journaled stream files

Applications using commitment control will see less improvement (commitment control already performs some journal caching).

Journal caching modifies the behavior of traditional noncached journaling in batch. Without journal caching, a batch job waits for each new journal entry to be written to disk. Journal caching allows most operations to no longer be held up waiting for synchronous disk writes to the journal receiver.

Journal caching is especially useful for situations where journaling is being used to enable replication to a second system.

It is not recommended to use journal caching 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.

» Furthermore, the results from the following commands or API will not display the journal entries in the cache :

- Display Journal (DSPJRN) command
- Retrieve Journal Entry (RTVJRNE) command
- Receive Journal Entry (RCVJRNE) command
- Retrieve Journal Entries (QjoRetrieveJournalEntries) API

The Display Journal Receiver Attributes (DSPJRNRCVA) Command and the Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API show the total number of journal entries in a journal receiver. However if some of those entries are in the cache, you cannot see these journal entries using the DSPJRN, RTVJRNE, and RCVJRNE commands, and the QjoRetrieveJournalEntries API. For example, if there are 100 journal entries in a journal receiver, the DSPJRNRCVA command and QjoRtvJrnReceiverInformation API show that the total number of entries is 100. However, if the last 25 entries are in the journal cache, you can only view the first 75 entries.

Journal caching also affects remote journaling. Journal entries are not sent to the remote system until they are written from the cache to disk. Since journal entries are not sent to the target system right away, the number of journal entries that are not confirmed are always greater than if you are not using journal caching.

«

Contact your service representative for more information about ordering journal caching.

Object assignment to journals: You can use one journal to manage all the objects you are journaling. Or, you can set up several journals if groups of objects have different backup and recovery requirements. Every journal has a single attached receiver. All journal entries for all objects being managed by the journal are written to the same journal receiver.

When deciding how many journals to use and how to assign objects to journals, consider the following:

- Using one journal (and journal receiver) is the simplest method for managing both daily operations and recovery.
- There is a limit of 250 000 objects that can be journaled to a single journal.
- If using a single journal receiver causes a performance bottleneck, you can alleviate this by placing the journal receiver in a separate disk pool from the objects that you are journaling.
- To simplify recovery, assign objects that are used together in the same application to the same journal.
- If you are journaling database files, all the physical files underlying a logical file must be assigned to the same journal.
- Files opened under the same commitment definition within a job can be journaled to different journals. In commitment control, each journal is considered a local location.
- If your major applications have completely separate objects and backup schedules, separate journals for the applications may simplify operating procedures and recovery.
- If you journal different objects for different reasons; such as recovery, auditing, or transferring transactions to another system; you may want to separate these functions into separate journals. However, you can assign an object to only one journal.
- If the security of certain objects requires that you exclude their backup and recovery procedures from the procedures for other objects, assign them to a separate journal, if possible.
- If you have basic disk pools with libraries, all objects assigned to a journal must be in the same disk pool as the journal. The journal receiver may be in a different disk pool. If you place a journal in a disk pool without libraries (non library disk pool), objects being journaled must be in the system disk pool. The journal receiver may be in either the system disk pool or the non library disk pool with the journal. See Determine the type disk pool in which to place journal receivers for more information about the types of disk pools.

- If you have independent disk pools, they must be library capable in order to journal objects on them. You cannot journal objects on User-Defined File System (UDFS) independent disk pools.

Set up journaling

The following gives instructions on how to set up journaling

After you have decided how you will use journaling, follow these steps to set up journaling on your system. If you have decided to use more than one journal, work through all the steps for one journal at a time. Then repeat the steps for the next journal.

You can choose one of the following methods to set up journaling:
- Set up journaling with iSeries(TM) Navigator.
- Set up journaling with the character-based interface.

See Example: Set up journaling for a code example of setting up journaling for character-based interface.

Note: Read the Code example disclaimer for important legal information.

For information about the difference between the two methods, see iSeries Navigator versus character-based interface for object journaling.

Information you need to set up journaling

Setting up journaling consists of creating a journal and a journal receiver, then starting journaling. When you create a journal receiver you have the following information:

Information to create the journal receiver
- The name of the journal receiver
- The disk pool assignment for journal receiver
- The storage threshold for the journal receiver
- Who has who has authority to the journal receiver

Information to create the journal
- The name of the journal
- The library assignment of the journal
- The journal receiver name to associate with the journal
- Which disk pool to assign storage space for the journal (only if you are using the ASP parameter in the CRTJRN command)
- The journal message queue
- Whether you will use manual or system journal-receiver management
- Whether or not to have automatic deletion of the journal receiver
- Receiver size options for the journal
- Who has authority to the journal
- Whether or not to minimize entry-specific data (character-based interface only)
- Whether or not to use journal caching (character-based interface only)
- Whether or not to delay the next attempt to automatically change the journal receiver (character-based interface only)
- Whether or not to delay the next attempt to automatically delete the journal receiver (character-based interface only)
- Whether or not to include fixed-length data in the journal entries (character-based interface only)

Set up journaling with the character-based interface

1. Create the journal receiver using the Create Journal Receiver (CRTJRNRCV) command.
2. Create the journal using the Create Journal (CRTJRN) command.
3. Start journaling for each object that you plan to journal.

Set up journaling with iSeries Navigator

1. Expand **Databases**.
2. Expand your system's local database.
3. Expand **Schemas**.
4. Right click the schema in which you want to create the journal.
5. Select **New->Journals**
6. Start journaling for each object that you plan to journal.

Example: Set up journaling

The following are three examples of setting up journaling in the character-based interface. The first example sets up journaling with the both the journal and receiver in the system disk pool. The second and third examples set up journaling with the journal and journal receiver in separate basic disk pools.

Note: Read the Code example disclaimer for important legal information.

Journal and receiver in system disk pool

In this example, the library $DSTJRN is in the system disk pool and has the following description:

- Type: PROD
- Disk pool of library: 1
- Create authority: *EXCLUDE

1. The $DSTJRN library already exists in the system disk pool.
2. The Create Journal Receiver (CRTJRNRCV) command creates journal receiver RCVDST1 in the $DSTJRN library: »

   ```
   CRTJRNRCV JRNRCV($DSTJRN/RCVDST1) THRESHOLD(1500000)
             TEXT('RECEIVER FOR $DSTJRN JOURNAL')
   ```

   «
3. The journal receiver is placed in the system disk pool with the library because *LIBASP is the default value for the ASP parameter on the CRTJRNRCV command.
4. Public authority for the journal receiver is *EXCLUDE because the **Create authority** value for the library is *EXCLUDE and the default for the authority (AUT) parameter is *LIBCRTAUT.
5. The Create Journal (CRTJRN) command creates the associated local journal: »

   ```
   CRTJRN JRN($DSTJRN/JRNLA) JRNRCV($DSTJRN/RCVDST1)
          MNGRCV(*USER)
   ```

   «

» The journal is placed in the system disk pool with the journal receiver. The user manages the journal receiver. The receiver size option is *MAXOPT1 since RCVRSIZOPT(*MAXOPT1) is the default for the CRTJRN command. «

Journal receiver in a nonlibrary basic disk pool

In this example, the journal receiver is in a nonlibrary basic disk pool and the journal is in the system disk pool.

1. The CRTJRNRCV command creates journal receiver RCVDST2 in a nonlibrary basic disk pool »

```
CRTJRNRCV JRNRCV($DSTJRN/RCVDST2) THRESHOLD(1000000)
          ASP(2) TEXT('RECEIVER FOR $DSTJRN JOURNAL')
```

«

2. The CRTJRN command creates the local journal in the system disk pool: »

   ```
   CRTJRN JRN($DSTJRN/JRNLB) JRNRCVR($DSTJRN/RCVDST2)
          MSGQ($DSTJRN/JRNLBMSG)
       MNGRCV(*USER)
   ```

   «

3. When the receiver RCVDST2 exceeds 1 024 000 000 bytes of storage, a message (CPF7099) is sent to the JRNLBMSG message queue in the $DSTJRN library.
4. The objects to be journaled must also be in the system disk pool.

Journal and journal receiver in basic disk pools

In this example, the libraries ARLIBR and ARLIB are in basic library disk pools and have the following description:

ARLIBR

- Type: PROD
- Disk pool of library: 3
- Create authority: *USE
- Text description: A/R Receiver LIB

ARLIB

- Type: PROD
- Disk pool of library: 4
- Create authority: *USE
- Text description: A/R Receiver LIB

1. »The CRTJRNRCV command creates journal receiver RCVDST3 in the library basic disk pool

   ```
   CRTJRNRCV JRNRCV(ARLIBR/RCVDST3) THRESHOLD(1500000)
             TEXT('RECEIVER FOR ARJRN JOURNAL')
   ```

   «

2. Because public authority is not specified, the public authority is set to *USE (the **Create authority** value for the ARLIBR library).
3. The CRTJRN command creates the local journal that is associated with the RCVDST3 journal receiver:

   ```
   CRTJRN JRN(ARLIB/ARJRN) JRNRCV(ARLIBR/RCVDST3)
   ```

   »When the RCVDST3 journal receiver exceeds 1 536 000 000 bytes of storage, the system creates a new journal receiver named RCVDST4, attaches it to the journal, and sends message CPF7020 (journal receiver detached) to the QSYSOPR message queue (the default queue).«
4. All objects journaled to the ARJRN journal must be in ASP 4 because the journal is in ASP 4.
5. »In this case, the database files and journal are in the same library. The journal receivers are in a library that is saved and restored after the journal library if a single command is used, because ARLIBR comes after ARLIB in a normal sort sequence.«

Start and end journaling and change journaling attributes

Following are instructions on how to start and end journaling for all of the object types that journaling supports.

Why you must save objects after you start journaling
After you start journaling, it is essential that you save objects that you are journaling.

Start journaling
This topic provides information about how to start journaling for all object types.

End journaling
This topic provides information about how to end journaling and why ending journaling might be necessary.

Change journaling attributes of journaled objects
This topic provides information about how to change the journaling attributes of a journaled object without ending journaling.

Why you must save objects after you start journaling

» It is critical to save the journaled object after journaling is started to be able to apply journaled changes. When you start journaling an object, the system assigns a unique **journal identifier** (JID) to that object. If the object is a physical database file, each member is also assigned a unique JID. If you start journaling on a distributed file, the piece on each server has its own unique JID. The JID is part of every journal entry added to the journal receiver for a given object. The system uses the JID to associate the journal entry with the corresponding journaled object. The copy of the object on the save media that was saved before it was journaled does not have the journal identifier saved with it. Therefore, if this copy of the object is restored to the server, the journal entries cannot be associated with the object and cannot be applied.

After you start journaling an object, do the following:

- Save the object immediately after you have started journaling it, before any changes have occurred.
- Save a physical file or a logical file after you start journaling access paths for the file. This ensures that when you restore the file, journaling access paths is started automatically.
- If you are using distributed files, save the file separately on the systems in the node group after starting journaling for the distributed file.

Saving these objects ensures that you can completely recover all the objects by using your saved copy and your journal receivers.

Update the history

If you are not using the save-while-active function, update the history for the object when you save it so that processing for applying and removing journaled changes will have the best information for verification. If you save the object using the SAV command, change the UPDHST value to something other than *NO. The default value for the SAV command is to not preserve the update history. For the other Save related commands, the default value is to preserve the update history. When you use the save-while-active function, you do not need to update the history for the object for verification when you apply and remove journaled changes. When you use the save-while-active function, information is saved on the media with the object and restored when the object is restored. This extra information provides the last save information for applying and removing journaled changes.

The JID and other journaling operations

Not only do you need the JID to apply journaled changes, other journaling operations use the JID. All formats, except the *TYPE1, *TYPE2, and *TYPE3 formats, for the Display Journal (DSPJRN), Receive Journal Entry (RCVJRNE), or Retrieve Journal Entry (RTVJRNE) command include the JID for the object. The JID is also included with the *TYPEPTR and *JRNENTFMT format for the RCVJRNE command, as well as the Retrieve Journal Entries (QjoRetrieveJournalEntries) API. You can use the Retrieve JID

Information (QJORJIDI) API to retrieve an object's name (for an object not in the integrated file system) or the file identifier (for an object in the integrated file system), if you know its JID.

Commands for saving objects

You can use one of the following commands to save journaled objects:

Physical database files, data areas, and data queues

- Save Changed Objects (SAVCHGOBJ) and specify OBJTYPE(*object-type) OBJJRN(*YES)
- Save Object (SAVOBJ)
- Save Library (SAVLIB)
- Save (SAV)

Integrated file system objects

- SAV

See the Manually saving parts of your server topic for more information about saving journaled objects.

Start journaling

After you have created the journal and journal receiver, you can start journaling. When journaling has been started for an object, the system writes journal entries for all changes to the object.

The start journal command must obtain an exclusive lock on the object. However, for database physical files and integrated file system objects, you can start journaling even if an object is open. The recommended procedure for starting journaling is:

1. Start journaling the object.
2. Save the object. If the object is open for changing, this will be a save-while-active type save.

If you are not using the save-while-active function, it is highly recommended that you update the history for the object when you save it so that processing for applying and removing journaled changes will have the best information for verification. If you saved the object using the SAV command, the default value is to not preserve the update history. Therefore, change the UPDHST value to something other than *NO. For the other save related commands, the default value is to preserve the update history. When using save-while-active, updating the history for the object is not needed for verification when applying and removing journaled changes. In this case, information is saved on media with the object, and restored when the object is restored. This extra information provides the last save information for applying and removing journaled changes.

The following provides instructions to start journaling for each object type:

- Journal database physical files (tables)
- Journal DB2(R) Multisystem objects
- Journal integrated file system objects
- Journal access paths
- Journaling data areas and data queues

Journal database physical files (tables): When you start journaling, a physical file (table) you specify whether you want after-images saved, or both before-images and after-images.

To reduce the number of journal entries, you can omit entries for open operations and close operations for the file. To omit open and close entries from being journaled, select the **Exclude open and close entries** in iSeries(TM) Navigator. Or you can Specify `OMTJRNE(*OPNCLO)` on the Start Journal Physical File (STRJRNPF) command. If you choose to omit open journal entries and close journal entries, be aware that:

- You cannot use the journal to audit who has accessed the file.
- You cannot apply or remove journal changes to open boundaries and close boundaries using the TOJOBO and TOJOBC parameters.

Start journaling for physical database files

1. In iSeries Navigator, expand the system with the object you want to journal.
2. Expand **Databases** and the database with the object you want to journal
3. Expand **Schemas** and select the schema with the object you want to journal.
4. Click **Tables**.
5. Right-click the table you want to journal and select **Journaling**.

Or you can use the STRJRNPF command to start journaling physical database files.

The DB2 Universal Database(TM) topic has complete information about database files.

Journal DB2(R) Multisystem files: When you successfully start journaling on a distributed file, the system distributes the start journal request to the other servers in the node group. All servers are attempted even if there is a failure at any one server. Once journaling is started on a server in the node group, it stays started even if there is a failure at any of the other servers.

The journal has to exist with the same name on all servers in the node group. The journal itself is not distributed, only the Start Journal Physical File (STRJRNPF) command.

The journal and its receiver are associated only with the changes made to the file on the one server. If you have two servers in the node group and a file is updated on both servers, the update on server A is only in server A’s journal and receiver and the update on system B is only in system B’s journal and receiver.

» The journal identifier (JID) is different on each piece of the distributed file. Each server piece has its own JID. This means that you cannot use the journal entries that are deposited on one server to apply or remove journaled changes to a different piece of the file on another server. «

Journal integrated file system objects: You can journal the following integrated file system objects if they are in the ″root″(/), QOpenSys, and user-defined file systems:

- Stream files (*STMF)
- Directories (*DIR)
- Symbolic links (*SYMLNK)

» When you use the SAV command to save an integrated file system object, the default is to not update the history information for the object. If you plan to apply journaled changes to the objects you are journaling, and you are not using the save-while-active function, specify to preserve the update history information about the SAV command. «

If you are journaling *DIR or *STMF objects, you can reduce the number of journal entries in the journal receiver. In iSeries(TM) Navigator, if you ensure that **Include open, close, and synchronization entries** is unselected (`OMTJRNE(*OPNCLOSYN)` on the Start Journal (STRJRN) command), you can omit entries for open operations, close operations, and force entries for the object. If you choose not to journal these entries be aware of the following:

- You cannot use the journal to audit who has accessed the object for opens, closes, and forces.
- You cannot apply journal changes to open boundaries and close boundaries using the TOJOBO and TOJOBC parameters.
- This option is only valid for *DIR and *STMF objects.

If you are journaling symbolic links, the system does not follow the symbolic link to determine what to journal. That is, the system only journals the actual symbolic link. If you want to journal the end object, you must journal the end object directly.

If you are journaling a directory and select **Journal new files and folders** in iSeries Navigator (INHERIT(*YES) on the STRJRN command), then objects created into that directory will be automatically journaled to the same journal. Therefore use caution because you can journal more objects than you realize. Also, even if this option is on, if an object is restored to the directory, it keeps the journaling attributes it had before the restore operation (when it was saved). For example, if you restore a stream file that is journaled to Journal X, but the directory you restore the stream file to is being journaled to Journal Y, the stream file will still be journaled to Journal X, even if the directory has the inherit option on.

Note: If you end journaling for an object and then rename that object in the same directory in which it currently resides, journaling is not started for the object, even if the directory has the inherit option on.

If you select **Current[(R)] folder and all subfolders** in iSeries Navigator (SUBTREE(*ALL) on the STRJRN command), journaling only starts on objects that exist in the subtree when the STRJRN command is executed. To start journaling on objects that are added to the subtree after this point you have three options:

- You can start journaling for each object after it is created.
- You can select **Journal new files and folders** (INHERIT option) on the original start journal request.
- » After journaling is started you can use the Change Journaled Objects (CHGJRNOBJ) command and specify INHERIT(*YES).«

If you select to journal the current folder and all subfolders, and there are object types in the subtree that are not supported for journaling, the unsupported object types are skipped over so that only object types that are supported for journaling get journaled.

Restrictions for journaling integrated file system objects

Restrictions for journaling integrated file system objects are as follows:

- You cannot journal files which are memory mapped. The Memory Map a File (mmap()) API documentation has information about memory mapping.
- iSeries servers allocate disk space for Integrated xSeries[(R)] servers as virtual disk drives for the xSeries servers. From the perspective of the iSeries server, virtual drives appear as byte stream files within the integrated file system. You cannot journal these byte stream files. See the Windows[(R)] environment on iSeries topic for more information about Integrated xSeries servers.

Start journaling for integrated file system objects

To start journaling for integrated file system objects do the following steps:

1. In iSeries Navigator select the system on which the object that you want to journal is located.
2. Expand File Systems.
3. Expand Integrated File Systems.
4. Expand the file system with the object you want to journal.
5. If you are journaling a directory, right-click the directory and select **Journaling**.
6. If you are journaling an object in a directory, expand the directory and right click that object. Select **Journaling**.

Or, use the STRJRN command or Start Journal (QjoStartJournal) API for integrated file system objects that you want to journal.

For more information about integrated file system objects, see the Integrated file system topic.

Journal access paths: After you have started journaling for physical files, you can set up explicit journaling of access paths. You can use the Start Journal Access Path (STRJRNAP) command to start journaling access paths owned by physical files or logical files. When you start journaling access paths for a physical file, the system journals any of these, if they exist:

- Keyed access paths
- Access paths for primary key constraints
- Access paths for unique constraints
- Access paths for referential constraints

» The system does not journal access paths that use an international component for unicode (ICU) sort sequence table or encoded vector access paths. «

All underlying physical files must be journaled to the same journal before you can start journaling for an access path. The entries created when you journal an access path are used to recover the access path after the system ends abnormally. They are not used when you apply or remove journal entries. You can specify RCVSIZOPT(*RMVINTENT) for the journal to have the system remove these entries when they are no longer needed for recovery. This reduces the disk storage requirements for the journal receiver.

You cannot start journaling for an access path that is in use. The STRJRNAP command must obtain an *EXCL lock on the logical file.

The recommended procedure for starting access path journaling is as follows:

1. Use the STRJRNAP command to start journaling the access path.
2. Save all the underlying physical files, specifying ACCPTH(*YES).

If you have target recovery times for access paths set up on your system, you may not need to set up explicit journaling for access paths. See Reasons to journal access paths for more information.

Journal data areas and data queues: When you start journaling for a data area or a data queue, the system writes journal entries for all changes to the data area or data queue.

When you start journaling a data area, you specify whether you want after-images saved, or both before-images and after-images.

Start journaling for data areas and data queues

1. In iSeries(TM) Navigator, expand the system with the data area or data queue you want to journal.
2. Expand File Systems.
3. Expand Integrated File System.
4. Expand QSYS.LIB.
5. Select the library with the data area or data queue.
6. Right-click the data area or data queue you want to journal and select **Journaling**.

Or after you have created the journal, use one the following commands or API for each data area or data queue you want to journal:

- Start Journal (STRJRN)
- Start Journal Object (STRJRNOBJ)
- Start Journal (QjoStartJournal) API

For more information about data queues, see CL programming For more information about data areas, see Work Management on the V5R1 Supplemental Manuals Web site.

Change journaling attributes of journaled objects without ending journaling

»

Use the Change Journaled Object (CHGJRNOBJ) command to change journaling attributes of journaled objects without ending and restarting journaling. You can use the CHGJRNOBJ command to do the following:

- Change whether you are journaling both before and after images or just after images.
- Change whether you are omitting open, close, and force journal entries from the journal receiver.
- Change whether you are journaling objects that are created in a directory.
- Remove the partial transaction state from a database file.

Except for removing the partial transaction state from a database file, the objects whose attributes you are changing must currently be journaled. Also, you can only change one attribute at a time.

Before and after images

Use the Images (IMAGES) parameter to change if you are journaling only after images or both before and after images. The object whose journaling attributes you are changing must already be journaled. You can change this journaling attribute for the following object types:

- Database physical files
- Data areas

Omit journal entries

Use the Omit Journal Entries (OMTJRNE) parameter to change whether to omit open, close, and force journal entries from the journal receiver. The object whose journaling attributes you are changing must already be journaled. You can change this journaling attribute for the following object types:

- Database physical files
- Integrated file system stream files
- Integrated file system directories

Journal new objects in a directory

Use the New Objects Inherit Journaling (INHERIT) parameter to change whether journaling starts automatically for objects that are created in a journaled integrated file system directory after the attribute is changed.

Partial transaction state

Attention: Use of this parameter can result in loss of data. Use this parameter only as a last resort, if the appropriate journal receivers are unavailable to do an apply or remove journaled changes operation.

Use the Partial Transactions (PTLTNS) parameter to allow an object that contains partial transactions to be used. You use this parameter only for one of the following reasons:

- You are unable to apply or remove the journaled changes to complete or remove the transactions because the journal receivers are unavailable.

- The object was involved in a rollback operation that was ended early and there is no saved version of the object to use.

Only use this parameter as a last resort because the partial transactions remain within the object.

For instructions for recovering objects in a partial transaction state, see Example: Recover objects with partial transactions.

Consideration for distributed files

When you successfully change the journal attributes for a distributed file, the system distributes the request to change a journal attribute to the other servers in the group. All servers are attempted even if there is a failure at any one server. When the journaling attribute has been changed on a server in the node group, it remains that way even if there is a failure at any of the other servers.

«

End journaling

You may need to end journaling for several reasons:

- If a journal is damaged and you need to delete it, you must first end journaling for all objects assigned to the journal.
- In some situations, you might want to end journaling before running a large batch application, if that application has exclusive use of the object. This is done either to improve the speed of the batch application or to reduce the auxiliary storage needed for the journal receiver. If you do this, use this method:
  1. End journaling for the objects.
  2. If journaling physical files save them specifying ACCPTH(*YES).
  3. If journaling other object types, save them.
  4. Run the batch application.
  5. Start journaling for the objects.
  6. Save the physical files, specifying ACCPTH(*YES).
  7. Save the other journaled objects.

To end journaling proceed as follows:

1. End journaling for access paths with the End Journal Access Path (ENDJRNAP) command
2. In iSeries(TM) Navigator expand the system with the object that you want stop journaling.
3. If the object is a database file, proceed as follows:
   a. Expand **Databases** and the database with the journal that you want to end journaling.
   b. Expand **Schemas**.
   c. Click the schema with the table (file) you are journaling.
   d. Click **Tables**.
   e. Right-click table and select **Journaling**.
   f. Click **End** to end journaling.
4. If the object is an integrated file system object proceed as follows:
   a. Expand **File Systems**.
   b. Expand **Integrated File System**.
   c. Expand the file system with the object you are ending journaling.
   d. If you are ending journaling for a directory, right click that directory. If you are ending journaling for an object in a directory, open the directory and right click the object.
   e. Right-click the object or directory and select **Journaling**

f. Click **End** to end journaling.

5. If the object is a data area or data queue, proceed as follows:
   a. Expand **File Systems**.
   b. Expand **Integrated File System**.
   c. Expand **QSYS.LIB**.
   d. Select the library with the data area or data queue.
   e. Right-click the data area or data queue you want to end journaling and select **Journaling**.
   f. Click **End** to end journaling.

Or, use the following commands or API to end journaling :

- End Journal Access Path (ENDJRNAP) command for access paths
- End Journal Physical File (ENDJRNPF) command for database files
- End Journal (ENDJRN) command for integrated file system objects
- End Journal Object (ENDJRNOBJ) command for other objects
- End Journal (QjoEndJournal) API for integrated file system objects, data areas, and data queues.

You must end journaling for any access paths based on a physical file before you can end journaling for the physical file.

In the following cases, the system implicitly ends journaling:

- When you delete an object, journaling is ended for the object.
- When you remove a physical file member, journaling is ended for the member.
- When you remove a physical file member, journaling is ended for any access paths associated with the member unless an access path is shared and journaled by another file member.
- When you delete a file, journaling is ended for any access paths associated with the file unless an access path is shared and journaled by another file.

How to end journaling for distributed files

When you successfully end journaling on a distributed file, the system distributes the end journal request to the other systems in the node group. All systems are attempted even if there is a failure at any one system. Once journaling is ended on a system in the node group, it stays ended even if there is a failure at any of the other systems.

Even if a distributed file is not locally journaled, and if you specify the file name and the journal name on the ENDJRNPF command, the system will still attempt to distribute the end-journal request to the other systems in the file node group.

Distributed database administration has more information about distributed files.

Manage journals

Managing your journaling environment requires these basic tasks:

- Keep records of which objects you are journaling.
- Evaluate the impact on journaling when new applications or logical files are added.
- Regularly detach, save, and delete journal receivers.

Your journal receivers enable you to recover changes to your important objects. They also provide an audit trail of activity that occurs on your system.

Protect your journal receivers by regularly detaching them and saving them; or you can have the system take over the job of changing journal receivers by specifying system journal-receiver management.

Do the following tasks to manage your journaling environment:

- Swap, delete, and save journals and receivers
- Evaluate how system changes affect journal management
- Keep records of journaled objects
- Manage security for journals
- Display information for journaled objects, journals, and receivers
- Work with inoperable journal receivers
- Compare journal images
- Work with IBM[(R)]-supplied journals
- Send your own journal entries
- Change the state of local journals

Swap, delete, and save journals and receivers

The management tasks that you need to perform most often for journaling are swapping journal receivers and saving and deleting journal receivers.

See the following information to accomplish these tasks:

- Swap journal receivers
- Keep track of journal receiver chains
- Reset the sequence number of journal entries
- Delete journal receivers
- Delete journals
- Save journals and journal receivers

Swap journal receivers: An important task for journal management is to swap (or change) journal receivers. You typically swap journal receivers when they reach their storage threshold. You can swap journal receivers either with iSeries[(TM)] Navigator or with the Change Journal (CHGJRN) command. If you use system journal-receiver management, the system changes journal receivers for you.

You can use iSeries Navigator or the Change Journal (CHGJRN) command to change the attributes of the journal. You also use the iSeries Navigator or the CHGJRN command to change the receiver for a journal (detach the current receiver, create and attach a new receiver) and to reset the sequence number for journal entries.

When you swap a journal receiver, the old journal receiver becomes detached. When you detach a journal receiver, you cannot reattached it to any journal. You can do these things with a detached journal receiver:

- Save or restore it.
- Display entries.
- Retrieve entries.
- Receive entries.
- Use it to apply or remove journaled changes.
- Use it to compare journaled images.
- Display its status or position in a receiver chain.
- Delete it.
- Replicate it with the remote journal function.

You must swap journal receivers to change the following journaling attributes:

- Manual or system journal management (MNGRCV parameter)

- Receiver size options (RCVSIZOPT parameter)
- Minimized entry specific data (MINENTDTA parameter)
- Fixed-length data (FIXLENDTA parameter)

To swap a journal receiver with iSeries Navigator, without changing options proceed as follows:

1. In the **iSeries Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database that you want to work with and **Schemas**.
4. Right-click the journal you want to use and select **Swap Receivers**. The system generates a new name when it creates the receiver.

To change options when you swap a journal receiver with iSeries Navigator proceed as follows:

1. In the **iSeries Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database that you want to work with and **Schemas**.
4. Double-click the journal you want to use.
5. Select **Swap receivers** and the journaling options you want to use.
6. Click **OK**. The **Journal properties** dialog closes. The new journal is automatically created and attached.

The CHGJRN command

Use JRNRCV(*GEN) on the Change Journal (CHGJRN) command to create the new receiver with the same attributes as the currently attached receiver, and in the same library. These attributes include the owner, private authorities, public authority, object auditing, ASP identifier, threshold, and text.

You must use the CHGJRN command to change the journaling options to one of the following:

- »Specify a journal receiver-size option.«
- Specify that objects allow journal entries to have minimized entry-specific data.
- Specify the data that is included in the fixed-length portion of the journal entries.
- Specify the time to delay the next attempt to automatically attach or delete a new journal receiver with system journal receiver management.
- Specify journal caching.
- Specify journal standby state.

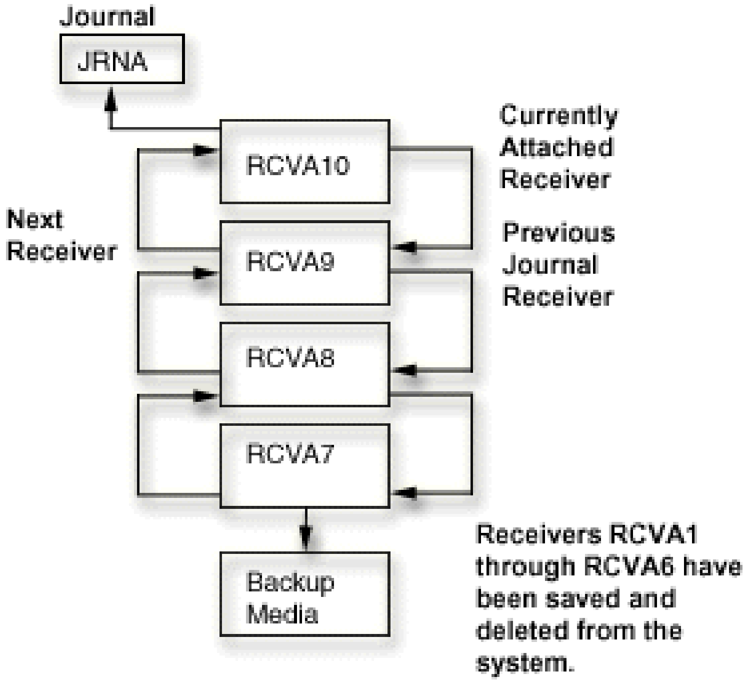
»**Caution:** If you use save-while-active operations to save objects before they reach a commitment boundary, ensure that you save the journal receiver after you detach it. If you delete the journal receiver before it is saved, you can lose the ability to recover any pending transactions for those objects. «

See Manual versus system journal-receiver management to help you decide if you to have the system change the journal receiver automatically. See Threshold (disk space) for journal receivers for more details about storage threshold. Save your server while it is active has instructions for saving an object with transactions in a partial state.

Keep track of journal receiver chains: Journal receivers that are associated with a journal (that is presently or previously attached to the journal) are linked in one or more **receiver chains**. Each journal receiver, except the first one, has a previous receiver that was detached when the current receiver was attached. Each journal receiver, except the one that is currently attached, also has a next receiver.

The following figure illustrates the process by which journal receiver chains are created. If you leave the previously attached receivers RCVA7 through RCVA9 online, you can use them to apply changes, to remove changes, or to display journal entries without restoring them first.

Journal receiver chain


Journal
JRNA
Currently Attached Receiver
RCVA10
Next Receiver
RCVA9
Previous Journal Receiver
RCVA8
RCVA7
Backup Media
Receivers RCVA1 through RCVA6 have been saved and deleted from the system.

If a complete copy of a receiver is missing in a chain of journal receivers linked together in the previously described relationship, the result is a **chain break**. Avoid receiver chain breaks. A receiver chain break indicates that any changes made between the last entry in the last receiver in one chain and the first entry in the first receiver in the next chain are not available in any journal receiver on the system.

»

Note: If you use save-while-active operations to save objects before they reach a commitment boundary, it is crucial that you keep track of your journal receiver chains.

Using a save-while-active operation to save objects before they reach a commitment boundary can result in objects saved to the media that have partial transactions. A break in a journal receiver chain can prevent you from recovering these objects with partial transactions. «

A set of receivers for a journal that has one or more receiver chain breaks has multiple receiver chains. Receiver chain breaks result from the following:

- You restored an old journal receiver and its next receiver is not on the system.
- A journal receiver was saved while it was attached, a partial receiver is restored, and no complete copy of the receiver is on the system or restored.
- A receiver that has not had its storage freed by a save operation is restored, and the next receiver has had its storage freed by a save operation.

- The journal is restored. All journal receivers associated with the previous copy of the journal (before the journal was deleted and restored) will not be in the same receiver chain as the currently attached journal receiver.
- The user or the system deleted a damaged or destroyed journal receiver from the middle of a chain.
- A journal receiver from another system is restored. The journal receiver will be associated with a journal at restore time if the associated library and journal on the source system had the same library name and journal name as the library and journal on the target system.
- You chose to replicate specific receivers instead of all receivers in the receiver directory chain. This occurred while replicating journal receivers from a source system to a target system.

You cannot use the following commands and API across multiple receiver chains:

- Apply Journaled Changes (APYJRNCHG)
- » Apply Journaled Changes Extend (APYJRNCHGX)«
- Remove Journaled Changes (RMVJRNCHG)
- Receive Journal Entries (RCVJRNE)
- Display Journal (DSPJRN)
- Retrieve Journal Entries (RTVJRNE)
- Compare Journal Images CMPJRNIMG
- Retrieve Journal Entries (QjoRetrieveJournalEntries) API

If multiple receiver chains exist, you need to determine:

- Whether any journal entries are missing.
- Whether your data will be valid if you use more than one receiver chain.

If you decide to proceed, you must run a separate command for each receiver chain.

You can use the Work with Journal Attributes (WRKJRNA) command to display the receiver chain (F15) and work with journal receivers. See Display information for journaled objects, journals, and receivers for more information about the WRKJRNA command.

Save your server while it is active has instructions for saving an object with transactions in a partial state. Example: Recover objects with partial transactions has instructions for recovering objects with transactions in a partial state.

Reset the sequence number of journal entries: Normally, when you change journal receivers, you continue the sequence number of journal entries. When the sequence number becomes very large, consider resetting the sequence to start the numbering at 1. You can reset the sequence number only when all changes are forced to auxiliary storage for all journaled objects and commitment control is not active for the journal. Resetting the sequence number has no effect on how the new journal receiver is named.

Some conditions prevent you from resetting the sequence number, such as an active commit cycle. If the system cannot reset the sequence number, you receive message CPF7018.

» Unless you specify RCVSIZOPT(*MAXOPT3) for the journal to which you attached the receiver, if you use system journal-receiver management for a journal, the sequence number for the journal is reset to 1 whenever you restart the system or vary on the independent disk pool containing the journal. When you restart the system or vary on an independent disk pool, the system performs the change journal operation for every journal on the system or disk pool that specifies system journal-receiver management. The operation that the system performs is equivalent to `CHGJRN JRN(xxx) JRNRCV(*GEN) SEQOPT(*RESET)`. The sequence number is not reset if journal entries exist that are needed for commitment control IPL recovery.

If you specify RCVSIZOPT(*MAXOPT1) or RCVSIZOPT(*MAXOPT2) for the journal to which you attached the receiver, the maximum sequence number is 9 999 999 999. If you specify RCVSIZOPT(*MAXOPT3), the maximum sequence number is 18 446 744 073 709 551 600. If you do not specify a receiver-size option, the maximum sequence number is 2 147 483 136. If these numbers are reached, journaling stops for that journal. Whenever you change journal receivers, the system tells you what the starting sequence number is through message CPF7019. Also, when you are approaching the sequence number limit, CPF7019 is additionally sent to the QSYSOPR message queue every time you change journal receivers.

The system sends a warning message (CPI70E7) to the journal's message queue when the sequence number exceeds 2 147 000 000. If you specified RCVSIZOPT(*MAXOPT1) or RCVSIZOPT(*MAXOPT2) for the journal that you attached the receiver to, the system sends the warning message when the sequence number exceeds 9 900 000 000. If you specified RCVSIZOPT(*MAXOPT3), the system sends the warning message when the sequence number exceeds 18 446 644 000 000 000 000. If you use system change-journal management support (MNGRCV(*SYSTEM)) for the journal, the system attempts to change the journal and reset the sequence number one time. The message is sent only if the attempt is not successful. «

To reset the sequence numbers for journal entries proceed as follows:

1. In the **iSeries(TM) Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database that you want to work with and **Schemas**.
4. Double-click the journal you want to use.
5. Select **Swap receivers** and under **Sequence numbering** select **Reset**.
6. Click **OK**. The **Journal properties** dialog closes. The new journal receiver is automatically created and attached.

Note: If you attempt to use the CHGJRN command with the same journal receiver name and SEQOPT(*CONT), you may receive the message CPF701A. To recover, delete the journal receiver and use the CHGJRN command again.

To change the sequence number with the Change Journal (CHGJRN) command, specify the SEQOPT(*RESET) parameter.

Delete journal receivers: Journal receivers can quickly use a lot of auxiliary storage space. Therefore an important journal management task is to delete journal receivers after you no longer need them.

How to determine whether to delete a journal receiver

When you are determining whether to delete a journal receiver, consider the following:

- Journal receivers you need for recovery
- Journal receivers you do not need for recovery
- Where the journal receiver is in the receiver chain

Journal receivers you need for recovery

» Do not delete a journal receiver that has not been saved if you need that journal for recovery. A journal receiver that you need for recovery is any journal receiver that you need to perform an apply or remove journaled changes operation.

Attention: Use care when you delete journal receivers if you use save-while-active operations to save objects before they reach a commitment boundary. Ensure that you save the journal receivers before you delete them. If an object is saved before it reaches a commitment boundary it can have partial transactions. If you need to restore objects with partial transactions, you must have access to the journal receivers that were attached during the partial transactions to avoid data loss.

«

To determine if a journal receiver has been saved, in iSeries(TM) Navigator, right-click the journal receiver, and select **Properties**. If the **Saved** field shows no date, then you have not saved the journal receiver.

If you have saved the journal receiver, but the journaled objects are not saved, then you still need that journal receiver for recovery. If you have space on your system, wait to delete journal receivers until it is unlikely that you need them for a recovery. (You saved the journaled object). Restoring journal receivers before applying or removing journaled changes may significantly increase your recovery time.

Although it is not recommended, the system does not prevent you from deleting a receiver you detached and is not saved or that is required to provide adequate recovery. If you try to delete a journal receiver that was once attached but has not been saved, the system issues an inquiry message. You can then continue or cancel the delete operation. You can use the system reply list to specify the reply the system is to send for this inquiry message (rather than explicitly responding to each inquiry message).

Journal receivers you do not need for a recovery

If you are journaling only for access path protection or use a for commitment control, you most likely you do not need the journal receivers to recover journaled changes. You do not need to save these journal receivers before deleting them.

To make your journaling tasks easier, you can even automate the deletion of these journal receivers by specifying the following:

- Specify system journal-receiver management
- Specify automatic deletion of journal receivers

When you specify automatic deletion of journal receivers, the system does not send a message when it deletes a journal receiver. By specifying automatic deletion for journal receivers, you indicate that do not need the journal receivers for user recovery.

Where the journal receiver is in the receiver chain

To ensure logical recovery, the system does not allow you to delete a journal receiver from the middle of the receiver chain unless one of the following conditions exists:

- The journal is using automatic deletion of journal receivers
- The journal is a remote journal

However, if a journal receiver is damaged, you can delete it from the middle of the chain. If an attached journal receiver is damaged, you must perform a change journal operation for the damaged receiver before you can delete it.

Rules for deleting journal receivers

The rules for deleting journal receivers are as follows:

- You cannot delete a journal receiver that is attached to a local journal. You must perform a change journal operation to detach a journal receiver before you delete it.
- You must delete journal receivers in the same order they were attached to a journal.
- You can delete a damaged or inoperable receiver regardless of the previous restriction. However, if an attached receiver is damaged, you must detach it before you delete it.
- You cannot delete a journal receiver that is attached to a remote journal if the remote journal has a journal state of active. If you attempt to delete a receiver that is attached to a remote journal, the system sends the inquiry message CPA705E. The results of the reply to the message are the same as those that occur with message CPA7025.

Procedure for deleting journal receivers

Proceed as follows:

1. In the **iSeries Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database that you want to work with and **Schemas**.
4. Click the schema you want to work with.
5. Right-click the journal receiver you want to delete and click **Delete**.
6. At the **Confirm Object Deletion** dialog click **Delete**.

You can also use the Delete Journal Receiver (DLTJRNRCV) command to delete journal receivers. If you use the DLTJRNRCV command, an exit point is available to use with an exit program to help automate journal receiver deletion.

Exit point for the DLTJRNRCV command

One example of using this exit point is a situation where your application is using the data in the journal receiver. The application is dependent on the journal receiver being present until your application processing is complete. By registering an exit program with the QIBM_QJO_DLT_JRNRCV exit point, the program will be called every time a journal receiver is deleted from the system. If your program determines that your application is not yet done with the receiver, it can indicate that the journal receiver is not eligible for deletion.

If you must delete the receiver regardless of what an exit program indicates, you can specify *IGNEXITPGM for the DLTOPT parameter on the DLTJRNRCV command. This parameter value requests that any user exit programs that are registered for QIBM_QJO_DLT_JRNRCV exit point be ignored.

You can also use the following values for the DLTOPT parameter:

***IGNTGTRCV**
Ignore target receiver. If you specify this value, the system does not verify that all remote journals that are associated with this journal, and are immediately downstream on a target system, have full copies of this journal receiver. The delete operation will continue, even if a remote journal does not have a full copy.

***IGNINQMSG**
Ignore inquiry message. Inquiry message CPA7025 will not be presented, even if this receiver has not been fully saved. Also, inquiry message CPA705E is not presented to the user even if the receiver is attached to a remote journal. The delete operation continues.

Delete journals: Each journal on the system causes additional time and resource to be used when you restart the system or vary on an independent disk pool after an abnormal end. If you no longer need a journal, you can delete it. The system does not allow you to delete a journal if any of the following conditions exist:

- You are journaling objects to it.
- Commitment control is active, and the journal is associated with a commitment definition.

Note: If you have certain types of referential constraints defined, the system starts commitment control if it is not already started. For example, if you have defined a cascaded delete constraint for an object, the system starts commitment control if you open the object for a delete operation. The default commitment definition that is created is active until the job ends.

- Any of the associated remote journals have a journal state *ACTIVE.

If you no longer need a journal and its associated receivers, perform the following steps:

1. Use the Work with Journal Attributes (WRKJRNA) command to determine the following:
   - Which objects are being journaled to this journal
   - Whether or not commitment control is active and the journal is associated with it.
2. If commitment control is active and the journal is associated with it, end commitment control with the End Commitment Control (ENDCMTCTL) command.
3. End journaling for all objects associated with the journal.
4. If any commitment definitions are active that use this journal as the default journal, use the ENDJOB command to end the jobs that are using the commitment definitions. This includes commitment control that is started because of a referential constraint.
5. If any remote journals have a journal state of *ACTIVE, inactivate them. See Inactivate the replication of journal entries to a remote journal for more information.
6. Delete the journal by doing the following:
   a. In the **iSeries(TM) Navigator** window, expand the system you want to use.
   b. Expand **Databases**.
   c. Expand the database that you want to work with and **Schemas**.
   d. Click the schema you want to work with.
   e. Right-click the journal you want to delete and click **Delete**.
   f. At the **Confirm Object Deletion** dialog click **Delete**.
7. Delete the journal receiver.

You can also use the Delete Journal (DLTJRN) command to delete the journal and the Delete Journal Receiver (DLTJRNRCV) command to delete the journal receiver.

Save journals and journal receivers: » You must save the journal receivers when they are no longer attached, so that you have all the journal entries saved.

Using a save-while-active operation to save objects before they reach a commitment boundary can result in objects that are saved with partial transactions. Saving journal receivers ensures that they are available to recover objects that are restored with partial transactions. «

When you save a journal receiver that is no longer attached, you can free storage. However, a journal receiver whose storage has been freed must be restored before you can use it for recovery.

The following topics provide examples of approaches you might take in detaching and saving journal receivers.

- Use SAVCHGOBJ to save journal receivers
- Methods to save journal receivers
- Correct order for restoration of journaled objects

»

Saving journals and journal receivers in the Back up your server topic provides more information about saving journals and journal receivers. Example: Recover objects with partial transactions has instructions for recovering objects with partial transactions. «

Note: Read the Code example disclaimer for important legal information.

Use SAVCHGOBJ to save journal receiver: »One technique for saving journal receivers is to use the Save Changed Object (SAVCHGOBJ) command. When you use the SAVCHGOBJ command to save journal receivers, ensure that you omit the attached journal receiver.

In the following example, all your journal receivers are in a library called RCVLIB. The currently attached journal receiver is MYJRCV0004.

```
SAVCHGOBJ OBJ(*ALL) OMITOBJ(MYJRCV0004) LIB(RCVLIB) OBJTYPE(*JRNRCV)
          DEV(media-device-name) ENDOPT(*LEAVE)
```

This example saves all journal receivers that have any new entries since the entire library was saved but omits the currently attached journal receiver MYJRCV0004.

A possible disadvantage to using the SAVCHGOBJ command to save journal receivers is that you can accidentally save the journal receivers that are currently attached. Those journal receivers are saved as partial receivers. If you need to do a recovery, you may need to handle the error condition that occurs when you attempt to restore the partial receiver over the receiver that is currently on the system and has not yet been saved. Also, partial journal receivers make tasks such as displaying entries and performing apply and remove journaled changes operations more difficult. Therefore you must avoid saving attached journal receivers. «

Note: Read the Code example disclaimer for important legal information.

Methods to save journal receivers: Following are three methods to save journal receivers. The first method saves journal receivers individually. The two other methods save the journal receiver automatically.

Save journal receivers individually

Use the Work with Journal Attributes (WRKJRNA) command to display the receiver directory for each journal. The receiver directory tells which journal receivers have not yet been saved. Then use the Save Object (SAVOBJ) command to save them.

The advantage to using this technique is that each journal receiver is saved only once. You will not have problems with duplicate names and partial receivers if you need to restore. The disadvantage to this technique is that it requires manual effort to determine the names of the journal receivers to be saved.

Save journal receivers by name - Automated method 1

You can use a combination of system journal-receiver management and a control language (CL) program to automate most journal management tasks. Do the following:

- Specify a threshold size for the journal receiver.
- Specify MNGRCV(*SYSTEM), DLTRCV(*NO), and a message queue for the journal.
- Use a CL program to monitor the journal message queue for the message (CPF7020) that indicates that the system has successfully detached the journal receiver.
- Your CL program can then save the receiver that was detached and optionally delete it.

Save journal receivers by name - Automated method 2

An alternate method of automatically saving journal receivers is to use a high level language program that uses the Retrieve Journal Information (QjoRetrieveJournalInformation) API. The program can use this API to determine the journal receiver directory and which receivers are not saved. The program can then save the journal receivers that are not marked as saved. You can set up this program to run on a regular basis or as part of normal processing.

See CL Programming for information about control language programming.

Correct order for restoration of journaled objects: You must restore journals and their associated objects in the correct order. For the system to automatically reestablish your journaling environment, restore objects in this sequence:

1. Journals
2. Based-on physical files
3. Dependent logical files
4. Other journaled object types
5. Journal receivers.

You can restore journal receivers at any point after you restore the journals. You do not need to restore them after the journaled objects.

When these objects are in the same library, the system restores them in the correct sequence. When these objects are in different libraries or directories, you must restore them in the correct sequence, or you must manually reestablish your journaling environment after the restore operation.

You can restore journal receivers in any sequence. After restoring them, use option 9 (Associate receivers with journal) from the Work with Journal (WRKJRN) command display to build the receiver chain in the correct sequence. You can also use Option 9 to build the receiver chain if you restore the journal after the journal receivers. The journal must be on the system for the receiver chain to be built.

If you restore journaled objects before restoring the journal, you must start journaling again.

Your journals and journal receivers can be in different libraries. If this is true, you must ensure that the library that will contain the journal receivers is on the system before restoring the journal. Ensuring this will also ensure that the journal receiver is created in the desired library, since a journal receiver is created when the journal is restored. Only the library needs to be on the system, not the journal receivers in that library. If you do not ensure this, you may need to create a journal receiver in the desired journal receiver library. You would then have to run the Change Journal (CHGJRN) command to attach the new receiver to your journal.

See Backup and Recovery for more information about restoring objects to your server.

Evaluate how system changes affect journal management

After you have established your journaling environment, you need to keep up with changes that occur on your system.

When you add new applications, evaluate whether to journal the objects.

If you use SMAPP, the system automatically considers new access paths when deciding how to meet your target recovery times for access paths.

Journaling places some limits on what changes you can make. For example:

- You cannot protect a logical file, either explicitly or with SMAPP, if the underlying physical files are journaled to different journals.

- You cannot move an object to a different disk pool from the disk pool of the library that contains its journal.

Keep records of journaled objects

You must always have a current list of objects that you are journaling and their assigned journals. Print a new list whenever you add or remove objects from the journal. Do this to print a list:

1. Type WRKJRN
2. Specify *ALL for both the **Journal** and **Library** fields
3. Press the Enter key twice.
4. Write down the names of all the journals or use the PRINT key for each panel of the display.
5. For each journal on the list that is used to journal objects, type WRKJRNA JRN(*library-name/journal-name*) OUTPUT(*PRINT).

Keep the lists with your most recent set of backup media that you used to save the entire system. You can also use the Retrieve Journal Information (QjoRetrieveJournalInformation) API to retrieve information about your journaling environment.

You might need this list for the following reasons:

- You need to recover your journaling environment; for example, if the journal is damaged or deleted. Although you can recover your journaling environment by restoring the objects, in many cases starting journaling for the objects is a quicker and safer method.
- You create new access paths. The system cannot protect access paths, either explicitly or by using SMAPP, if the underlying physical files are not journaled to the same journal.
- You want to move objects to another disk pool. Journaled objects must be in the same disk pool as the journal, unless the objects are in the system disk pool and the journal is in a nonlibrary basic disk pool.

Keep records of your journal receivers

Choose the method for saving journal receivers that works best for your organization. Then be sure to keep track of what you do. Label your save media so that you know which journal receiver media volumes are required to apply journal changes to the last complete saved copy of the journaled objects.

Think through possible recovery scenarios. For example, assume this is your save procedure:

- You save all user libraries and directories on Sunday evening.
- You save changed objects every evening.
- You save journal receivers every 2 hours during normal business hours.

Given the preceding list, what are your recovery steps if you lose a journaled object at 3 p.m. on Thursday?

For complete information about developing a recovery plan, see Plan a backup and recovery strategy.

Manage security for journals

You can use journal management to provide an audit trail of changes that were made to your objects. You can determine which program or user made changes to objects by using the journal entries.

By specifying the FIXLENDTA parameter of the Change Journal (CHGJRN) or Create Journal (CRTJRN) commands you can specify that the following data is included in the journal entry:

- The job name.
- The effective user profile name.
- The program name.

- The program library name and the auxiliary storage pool device name that contains the program library.
- The system sequence number. The system sequence number gives a relative sequence to all journal entries in all journal receivers on the system.
- The remote address, the address family and the remote port.
- The thread identifier. The thread identifier helps distinguish between multiple threads running in the same job.
- The logical unit of work identifier. The logical unit of work identifies work related to specific commit cycles.
- The transaction identifier. The transaction identifier identifies transactions related to specific commit cycles.

For database physical files, you can determine what changes were made to specific records by using the Compare Journal Images (CMPJRNIMG) command. However, you cannot use the CMPJRNIMG command for journal entries that have minimized entry-specific data. If you specified the MINENTDTA(*FILE) parameter on the Create Journal (CRTJRN) or Change Journal (CHGJRN) commands, you might have minimized entry-specific data.

Use Journal management to provide an audit trail because of the following reasons:

- No one, even the security officer, can remove or change the journal entries.
- Journal entries represent a chronological sequence of events.
- Each journal entry in the system is sequentially numbered without gaps until the CHGJRN command resets the sequence number. A journal entry is written if the sequence number is reset.

Note: When you display the journal entries, there can be gaps in the sequence numbers because some journal entries are only used internally by the system. These gaps occur if you are using commitment control, database file journaling, or access-path journaling. To view the entries in the gaps, you can use the INCHIDENT parameter on the Display Journal (DSPJRN) command.

- The journal contains entries that indicate when each journal receiver was changed and the name of the next journal receiver in the chain.
- Whenever journaling for an object is ended or whenever an object is restored an entry is written.

Remember that the date and time recorded in the journal entries depends on the date and time entered during an IPL and therefore, may not represent the actual date and time. Also, if you use shared files, the program name that appears in the journal entry is the name of the program that first opened the shared file.

A special journal, that is called the audit (QAUDJRN) journal, can provide a record of many security-relevant events that occur on the system. See the iSeries(TM) Security Reference for information about the QAUDJRN journal.

For more information about security on your iSeries server, see the Security topic.

Display information for journaled objects, journals, and receivers

iSeries(TM) Navigator, Control Language commands, and APIs provide several ways you can display information about journaled objects, journals, and journal receivers.

Information for journaled objects

Use the following methods to get information about journaled objects.

iSeries Navigator
You can use iSeries Navigator to display information such as whether the object is journaled, the name of the object's journal, what library the object's journal is in, and which journaling options are being used. You can use iSeries Navigator to display journaling information for the following object types:

- Tables (database files)
- Integrated file system directories
- Integrated file system stream files
- Integrated file system symbolic links

CL commands and APIs

The advantage to using these commands and APIs is that they can get information about groups of objects. Using iSeries Navigator, you can only get information about one object at time. Use the following commands and APIs to get information about journaled objects.

- Display File Description (DSPFD)
- Display Object Description (DSPOBJD)
- Display Object Links (DSPLNK)
- Get Attributes (Qp0lGetAttr())
- List Objects (QUSLOBJ) API
- Open List of Objects (QGYOLOBJ)
- Work with Object Links (WRKLNK) «

Information for journal receivers

Ways that you can display information about journals and related receivers are as follows:

- iSeries Navigator
- Display Journal Receiver Attributes (DSPJRNRCVA) command
- Retrieve Journal Information (QjoRetrieveJournalInformation) API
- Work with Journal Attributes (WRKJRNA) command
- Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API

These methods can identify:

- The journal receivers currently attached to the journal
- A directory of the journal receivers still on the system that are associated with the journal.
- The names of all of the objects that are being journaled through the journal.
- The commitment control uses of this journal.
- The attributes of the journal.
- Information about all remote journals that are associated with the journal.

Furthermore, the DSPJRNRCVA command or the QjoRtvJrnReceiverInformation API can identify:

- Fixed-length data
- ASP of the journal receiver
- Minimized entry data
- The next and previous journal receiver information

You can find the status of a journal receiver by using the WRKJRNA command, then pressing F15 (Receiver directory) from the Work with Journal Attributes display. You can also use the DSPJRNRCVA command. Or in iSeries Navigator, you can the find status of a journal receiver by doing the following steps:

1. Expand the system with the journal receiver
2. Expand **Databases** and the database with the journal receiver.
3. Expand **Schemas** and the schema (library) with the journal receiver.
4. Click **Journal Receivers**.
5. Right-click the journal receiver, and select **Properties**.

When the journal receiver is in partial status

The **partial** status of a journal receiver indicates the following:

- The disk unit on which the journal receiver is stored is damaged. No more journal entries can be recorded.
- A journal receiver was saved while it was attached to the journal. This means that additional entries may have been recorded in the journal receiver after the save operation occurred. The receiver was later restored, and no complete version is available.
- The journal receiver is associated with a remote journal. It does not contain all the journal entries that are in the associated journal receiver that is attached to the source journal.
- A partial receiver does not contain all the entries that are recorded in the journal while this receiver was attached. It does contain entries that are recorded up to the last save operation.
- The most complete version of the journal receiver is no longer on the system because it was destroyed during a failure.
- You have restored an older, partial version.

Work with inoperable journal receivers

If you have specified journaling for any objects, the system ensures that you have corrected problems that affect journaling before continuing with operations on those objects. If the attached journal receiver becomes inoperable, the operation that writes a journal entry is interrupted and the system sends an inquiry message that notifies the system operator. The operator can swap the journal receiver with iSeries(TM) Navigator or the Change Journal (CHGJRN) command. You can then respond to the inquiry message. A receiver can become inoperable if the receiver is damaged, the maximum sequence number has been reached, or there is no more space.

Compare journal images

Use the Compare Journal Images (CMPJRNIMG) command to compare and list the differences between the before-image of a record and the after-image of that record, or the after-image of a record with the previous after-image of that record.

Note: If you are using maximum receiver-size option RCVSIZOPT(*MAXOPT3) and your entry sequence numbers exceed 9 999 999 999, specify the FROMENTLRG and TOENTLRG parameters when you use the CMPJRNIMG command.

You can only use the CMPJRNIMG command for journaled physical database files. You cannot use the CMPJRNIMG command for journal entries that have minimized entry-specific data. If you specified the minimized entry-specific data (MINENTDTA(*FILE) parameter on the Create Journal (CRTJRN) or Change Journal (CHGJRN) commands, the journal entries might have minimized entry-specific data, preventing you from being able to compare journaled images.

If the journaled files have null-capable fields, the null value indicators corresponding to the fields in the before-image of the record are compared with the null value indicators corresponding to the fields in the after-image of the record. A field-by-field basis compare does this.

The printed output from the CMPJRNIMG command shows the before-images and after-images of a record followed by a line that indicates (with asterisks) the specific change in the record on a character-by-character basis. If you compare the after-images, the output shows the previous after-image of the record and the current after-image of the record, followed by a line indicating the changes.

If you use this command to compare journal images for a file that contains any fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object), these fields are not included in the comparison. All other fields in the file are compared.

Work with IBM(R)-supplied journals

The operating system and some licensed programs use journals to provide audit trails and assist with recovery. The following table lists some of the IBM-supplied journals:

Journal name	Library name	Description
QACGJRN	QSYS	Keeps job accounting information. Work Management on the V5R1 Supplemental Manuals Web site describes the use of this optional journal.
QAOSDIAJRN	QUSRSYS	Provides recovery for the document library files and the distribution files. Used by Integrated xSeries(R) Server.
» QASOSCFG	QUSRSYS	The journal for the QASOSCFG physical file. The QASOSCFG file stores secure client SOCKets Secure (SOCKS) configuration data. The Client SOCKS support topic provides more information about SOCKS. «
QAUDJRN	QSYS	Keeps an audit record of security-relevant activity on the system. iSeries(TM) Security Reference describes this optional journal.
QCQJMJRN	QUSRSYS	Provides an audit trail for Managed System Services. See the Systems Management Journal section of the Managed System Services manual for more information about Managed System Services.
QDSNX	QUSRSYS	Provides an audit trail for DSNX activity.
» QIPFILTER	QUSRSYS	Provides information for troubleshooting and auditing IP filter rules. See the IP filtering and network address translation topic for more information about IP filtering rules. «
» QIPNAT	QUSRSYS	Provides information for troubleshooting and auditing network address translation (NAT). See the IP filtering and network address translation topic for more information about NAT. «
QLYJRN	QUSRSYS	Keeps a log of transactions made to the Application Development Manager datastore files.
QLYPRJLOG	QUSRSYS	Keeps the project logs for the Application Development Manager licensed program. Used by the system if recovery is necessary.
QLZALOG	QUSRSYS	Used by the licensed management program to log requests that exceed the usage limit of a license.

Journal name	Library name	Description
QPFRADJ	QSYS	Keeps a log of dynamic performance tuning information. Work Management on the V5R1 Supplemental Manuals Web site describes using this optional journal.
QSNADS	QUSRSYS	Provides an audit trail for SNADS activity.
» QSZAIR	QUSRSYS	A journal for Storage Management Services (SMS) «
QSNMP	QUSRSYS	Provides an audit trail for network management information. Simple Network Management Protocol (SNMP) Support describes using this journal.
QSXJRN	QUSRSYS	Provides a log of the activity that occurs in the database files for service-related activity. Keep the information in this journal for 30 days.
» QTOVDBJRN	QUSRSYS	A journal for virtual private networking (VPN). «
QVPN0001	QUSRSYS	Provides an audit trail for Virtual Private Networking (VPN) connections. TCP/IP Configuration and Reference describes this journal.
» QYPSDBJRN	QUSRSYS	A journal for the systems management platform «
QZCAJRN	QUSRSYS	Contains a record for each SNMP PDU in and out of the SNMP agent, by PDU type (SNMP GET, SNMP GETNEXT, SNMP SET$^{(TM)}$, SNMP TRAP). TCP/IP Configuration and Reference provides more information about this journal.
QZMF	QUSRSYS	Provides an audit trail for the mail server framework. AnyMail/400 Mail Server Framework Support provides more information about this journal.

If you are using licensed programs or system functions that require these journals, consult the documentation for those functions for instructions on how to manage the journals and journal receivers.

In general, you swap journal receivers to detach the journal receiver and create and attach a new receiver on a regular basis. You may need to save detached receivers before deleting them, or you may be able to delete them without saving them. This depends on how the journal receivers are being used and whether the journal is using system journal-receiver management.

In some cases, you can use the automatic cleanup function of Operational Assistant to remove detached journal receivers that are no longer needed.

Send your own journal entries

Use the Send Journal Entry (SNDJRNE) command or the Send Journal Entry (QJOSJRNE) API to add your own entries to a journal. The system places these entries in the journal's attached journal receiver along with the system-created journal entries.

To help identify your entries, you can associate each entry with a particular journaled object. If you use the QJOSJRNE API, you can include the commit cycle identifier with the journal entry and send a larger amount of entry-specific data.

You may add entries to the journal to identify significant events (such as a checkpoint) or to help in the recovery of your applications. On the SNDJRNE command, the data specified on the ENTDTA parameter becomes the **Entry-Specific Data** field in the journal entry, and the TYPE parameter value becomes the **entry type** field. On the QJOSJRNE API, you use the entry data parameter to specify the entry-specific data and the journal entry type parameter to specify the entry type. For both the command and API deposits, the entries journal code is ’U’.

Change the state of local journals

» Local journals can be in one of two states, active or standby. When the journal state of a local journal is active, journal entries are allowed to be deposited to the journal entry.

To put a journal in standby state use the Change Journal (CHGJRN) command. You can also use iSeries(TM) Navigator or the CHGJRN command to change the state of a journal back to active.

»

Standby state

Journal standby state is a separately purchased feature that prevents most journal entries from being deposited into the journal. Standby state is one of the features enabled via option 42 of the i5/OS(TM) operating system. You can start or end journaling objects while the journal is in standby. However, while a journal is in standby state, you cannot use explicit commitment control. Also, records within database files that have referential integrity constraints cannot be modified when the underlying journal is in standby state unless RESTRICT is specified on the ON UPDATE or ON DELETE attribute for the constraint. Additionally, records within database files that have data links defined cannot be modified when the underlying journal is in standby state.

An example of when you might want to put a journal into standby state is if the journal is on a backup system and you want the replicated copies of your objects on that system to incur very low overhead until role swap time. By having the journal in standby state until role swap time, a switchover to the target system can be accomplished more quickly because all objects on the backup system can remain journaled thus allowing the switchover processing to skip the costly step of starting journaling for all objects. Until the journal leaves standby state and reverts to active state the backup system is not incurring the overhead of journaling because most journal entries are not deposited when the journal is in standby state.

If there is an attempt to deposit a journal entry when the journal is in standby state, no entry is deposited, nor are any error messages sent to the application. In order to flag the transitions in and out of standby state, journal codes ’J’ and entry types ’SI’ and ’SX’ are deposited when the local journal is put into and out of standby. Even though the journal state is standby, and most journal entries are not deposited, there are a few critical journal entries that will be deposited in a journal. Use the Journal entry information finder to see if a journal entry is still deposited even though the journal is in standby state.

Additionally, when a journal is in standby state the system elects not to provide System-Managed Access-Path Protection (SMAPP) for any access paths built over files journaled to the journal and flags the access paths as not eligible for SMAPP protection. These access paths remain not eligible until the underlying journal leaves standby state and reverts to active state. Because the access paths are not eligible for protection, in some select instances system performance may be negatively impacted when a journal is changed to standby state, This would most likely occur if the access paths are large and are actively being changed. Under those conditions the underlying SMAPP mechanism attempts to compensate by enabling SMAPP protection for multiple small access paths whose keys are changing and whose underlying physical files are not associated with journals in standby state.

Also, abnormal IPL duration or the vary on of an independent Auxiliary Storage Pool (ASP) duration may be affected if standby state is chosen because some access paths that are no longer eligible for protection may need to be rebuilt.

If performance degrades after switching to standby state, then some investigation should be done to determine if standby state is a primary contributing factor. To reduce any potential performance impact, INCACCPTH(*ELIGIBLE) can be specified on the Change Recovery for Access Paths (CHGRCYAP) command. Specifying INCACCPTH(*ELIGIBLE) will reduce potential overhead but will expose you to a potentially longer IPL or vary on of an independent ASP. As with many other options, deciding to use standby state is a trade off between run time performance and IPL or independent ASP vary on duration.

To ensure that switching to standby state is not causing undo IPL or independent ASP vary on concerns, use the Display Recovery for Access paths (DSPRCYAP) command periodically to display the estimated access path recovery time. If this value is much larger than the target access path recovery time and the total not eligible recovery time is greater than zero, then use F13 (Display Not Eligible Access Paths) to display a list of the not eligible access paths. This will identify the access paths not eligible for SMAPP protection along with a reason for their not eligible status. If the access paths with the highest estimated rebuild times are not eligible due to standby, then you may wish to reconsider your standby choice. In lieu of standby, you may want to consider journal caching, which often provides nearly as much performance relief.

«

Active state

When a local journal is created, the journal state of that journal is *ACTIVE. This means that journal entries can be deposited to the local journal. If a local journal is in standby state, journal entries with journal code ’J’ and entry type ’LA’ are deposited when the local journal is activated.

If a local journal has been put in standby state, activate it by doing the following:

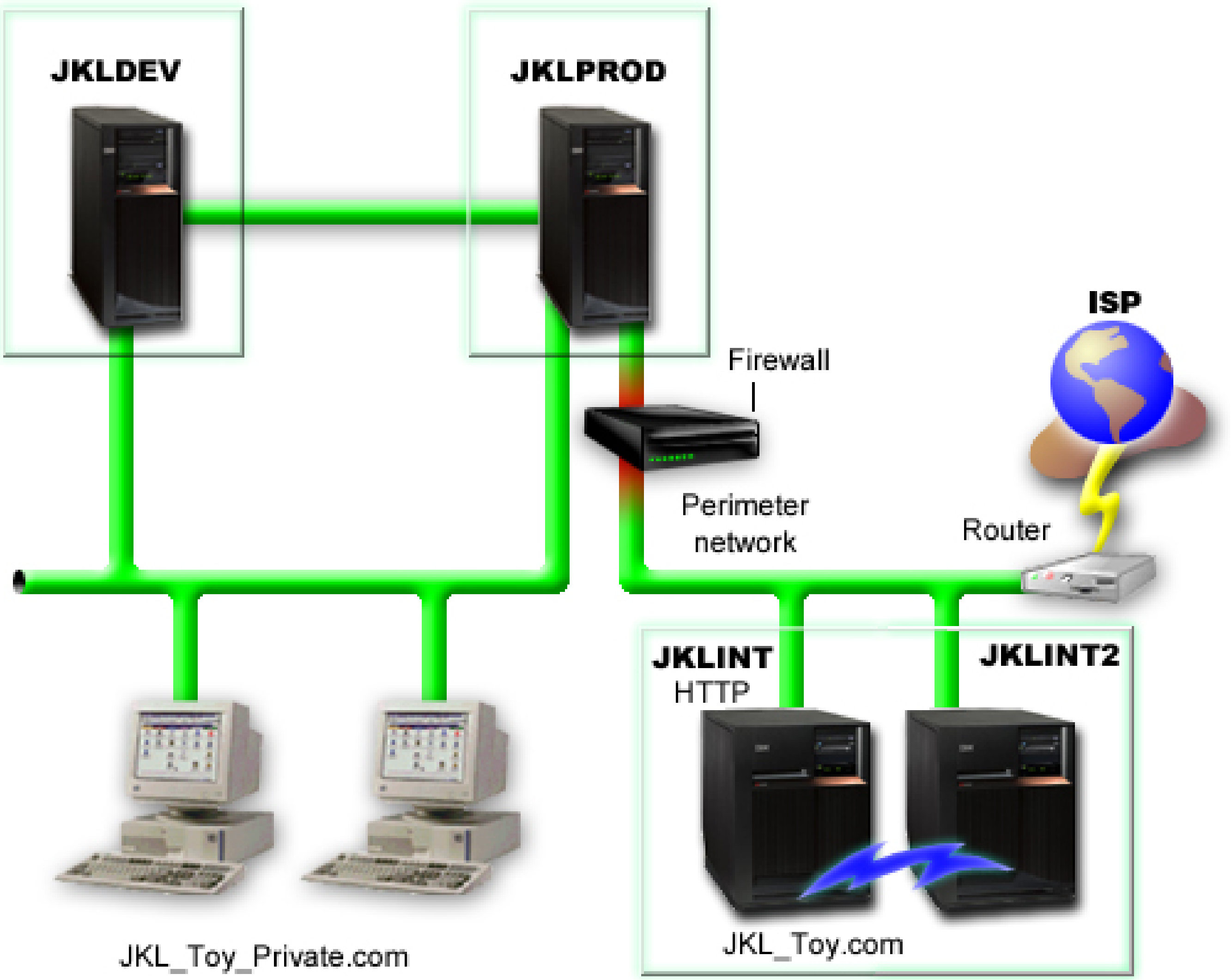
1. In the **iSeries Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database you want to work with and **Schemas**.
4. Click the Schema that contains the journal you want to activate.
5. Right-click the journal, and select **Properties**.
6. On the **Journal Properties** dialog box select **Activate journal**

You can also use the Change Journal State (QjoChangeJournalState) API or Change Journal (CHGJRN) command to activate the local journal. «

Scenario: Journal management

Sharon Jones, the system administrator for the JKL Toy Company, is responsible for backing up their servers and making sure that their servers can be recovered in the event of a natural disaster or system failure. As security officer, she is also responsible for ensuring the security of the servers.

The JKL Toy Company has a network that consists of a development server, a production server, and an http server. Click on a server on the diagram below for a description of the system and the journaling strategy Sharon uses.



JKLPROD

JKLPROD is the system that JKL uses for all of their customer orders and where their business applications are installed (inventory control, customer orders, contracts and pricing, accounts receivable). The information about this server is extremely critical to their business and changes often.

Also, there are several users who have remote access to the system from home connection. In addition, even though the company's web site is static, the company has plans to establish a transactional site. Because of the importance of the information about JKLPROD, Sharon wants to be able to audit the activity that occurs on the system.

JKLPROD journaling strategy

Since the objects on JKLPROD are crucial to JKL, and since they change often, Sharon has decided that they are good candidates for journaling.

- Since there are access paths that are critical to her operation, Sharon journals access paths.
- Sharon already separates the information about JKLPROD on separate disk pools:
  - Disk pool 2 - inventory control
  - Disk pool 3 - customer orders
  - Disk pool 4 - contracts and pricing

  - Disk pool 5 - accounts receivable

  Since the journal and the journaled objects must be in the same disk pool, Sharon creates four journals.
- Since she wants to audit the activity that occurs on the system, and since people have remote access to the system, Sharon journals fixed-length data using the following values:
  - Job name (*JOB)
  - User profile (*USR)
  - Program name (*PGM)
  - Remote address (*RMTADR)
- Since Sharon is using the FIXLENDTA parameter, she cannot minimize the fixed-length portion of the journal entries.
- Because she is using the FIXLENDTA parameter for all of the journals, and since she is journaling access paths Sharon uses the character-based interface to set up journaling.

Back to the scenario.

JKLINT

JKLINT is the system that JKL uses for their Web site and e-mail. While this data is critical to their business, it is fairly static.

They need 24x7 availability for the critical data on this server, and they accomplish that by having a second server, JKLINT2, that shadows JKLINT. They use a high availability replication solution to copy the data from JKLINT to JKLINT2. Then, if JKLINT goes down, they can switch to JKLINT2.

Since Sharon is using a high availability solution she uses remote journaling with the two servers. Scenarios: Remote journal management and recovery description shows the different ways that Sharon can set up remote journaling between JKLINT and JKLINT2.

Back to the scenario.

JKLDEV

JKLDEV is JKL's development server. Though it does not require 24x7 availability, the data on it represent many person hours of work by the developers. Therefore it is important that in the event of a crash, the system be brought to a current state. Also, since it is a development server, changes to the data occur often.

JKLDEV is used by both web and database developers. So several different types of data are stored on this server, including stream files and database files.

JKLDEV journaling strategy

Since many of the objects on JKLDEV are important and changes often, Sharon has decided that they are good candidates for journaling.

JKLDEV is used by both web and database developers, so there are several physical files, and many stream files that she wants to journal. Sharon has decided to do the following:
- Since none of the access paths are critical to her operation, Sharon does not journal access paths.
- To simplify setup and recovery, Sharon assigns all of the objects to one journal.
- Since there are many stream files to journal, Sharon journals the integrated file system directories, in addition to individual files. She elects to use the **Current**[(R)] **folder and all subfolders** option and **Journal new files and folders** option. This choice ensures that the objects currently in the directory and in any subfolders are journaled and objects that are created in the future are also journaled.
- Since journaling with the **Journal new files and folders** option can quickly make the journal receiver size grow quickly, she uses system journal-receiver management.

- Because it supports all of the options she has chosen, Sharon sets up journaling on iSeries(TM) Navigator.

Back to the scenario.

Recovery operations for journal management

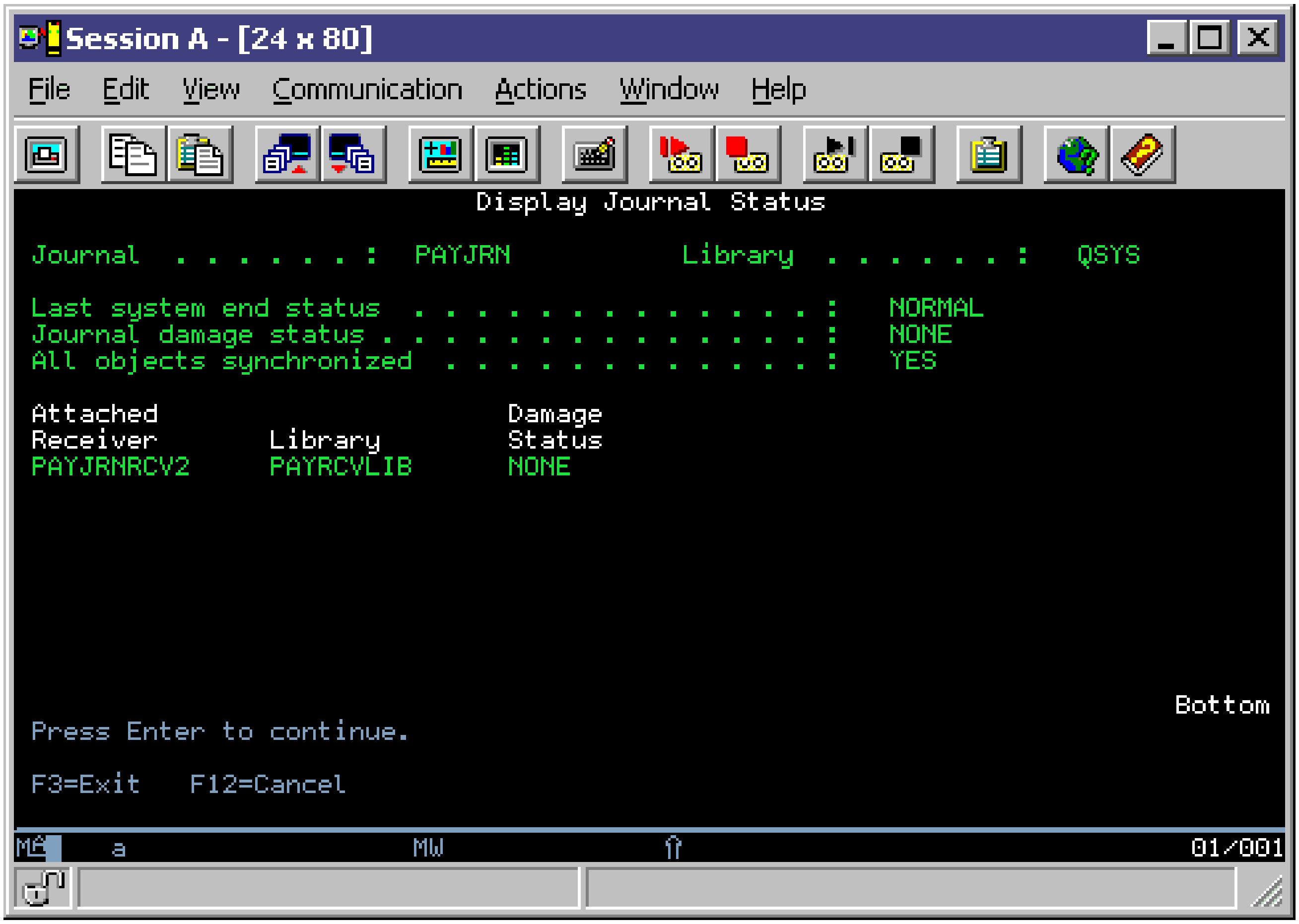
The following information contains recovery tasks to perform if you have an abnormal system end, need to recover a damaged journal, journal receiver, or journaled object:

- Determine recovery needs using journal status
- Recovery for journal management after abnormal system end
- Recover a damaged journal receiver
- Recover a damaged journal
- Recover journaled objects

Determine recovery needs using journal status

You can use the Work with Journal (WRKJRN) command to display the damage status of a journal and display whether or not the last IPL was normal.

Option 5 on the Work with Journal display shows the current status of the journal. It shows if the last system end was NORMAL or ABNORMAL, and if the journal is damaged. The damage status is NONE or FULL.



```
                           Display Journal Status

 Journal . . . . . . :   PAYJRN           Library . . . . . . :   QSYS

 Last system end status  . . . . . . . . . . . . . . :   NORMAL
 Journal damage status . . . . . . . . . . . . . . . :   NONE
 All objects synchronized  . . . . . . . . . . . . . :   YES

 Attached                                Damage
 Receiver         Library                Status
 PAYJRNRCV2       PAYRCVLIB              NONE










                                                                        Bottom
 Press Enter to continue.

 F3=Exit    F12=Cancel
```

If the last system end was abnormal, this display indicates whether the system synchronized the journaled objects or not. This indicates if the system synchronized each object in use during the abnormal end to match the entries in the attached journal receiver during the previous initial program load (IPL) or vary on of an independent disk pool.

If the last system end was normal, the display indicates that all objects are synchronized with the journal. If the journal is damaged, the display indicates that the system was unable to determine whether or not all objects are synchronized.

The display also presents information about the currently attached receiver and its damage status. The damage status of the receiver can be NONE, PARTIAL, or FULL. If the journal damage is such that the system cannot determine the status of the attached journal receiver, no attached receiver shows on the display.

If some objects are not synchronized or damage has been detected, a message appears indicating the form of recovery that you must perform.

Recovery for journal management after abnormal system end

If the system abnormally ends while you are journaling objects, the system does the following:

1. Brings all journals, journal receivers, and objects you are journaling to a usable and predictable condition during the IPL or vary on of an independent disk pool, including any access paths being journaled and in use at the time the system abnormally ended.
2. Checks all recently recorded entries in the journal receivers that were attached to a journal.
3. Places an entry in the journal to indicate that an abnormal system end occurred. When the system completes the IPL or vary on of an independent disk pool, all entries are available for processing.
4. Checks that the journal receivers attached to journals can be used for normal processing of the journal entries. If some of the objects you are journaling could not be synchronized with the journal, the system sends message CPF3172 to the history log (QHST) that identifies the journals that could not be synchronized. If a journal or a journal receiver is damaged, the system sends a message to the history log identifying the damage that occurred (message CPF3171 indicates that the journal is damaged, and messages CPF3173 or CPF3174 indicate that the journal receiver is damaged).
5. Recovers each object that was in use at the time the system ended abnormally, using the normal system recovery procedures for objects.

   In addition, if an object being journaled was opened for output, update, or delete operations, the system performs the following functions so changes to that object will not be lost:

   a. Ensures that the changes appear in the object. Changes that do not appear in the journal receiver are not in the object.
   b. Places an entry in the journal receiver that indicates whether the object was synchronized with the journal. For database files, if the file could not be synchronized with the journal, the system places message CPF3175 in the history log identifying the failure, and you must correct the problem. For other journaled objects, the system places message CPF700C in the history log identifying the failure, and you must correct the problem.

   A synchronization failure can occur if the data portion of the object is damaged, a journal receiver required to perform the synchronization is damaged, or the journal is inoperable.

Recover after abnormal system end

After an abnormal system end, perform the following steps:

1. Perform a manual IPL.
2. Check the history log to determine if there are any damaged objects, objects that are not synchronized, or any damaged journals or journal receivers.
3. If necessary, recover the damaged journals or journal receivers as described in Recover a damaged journal receiver and Recover a damaged journal.

4. If there is a damaged object:
   a. Delete the object.
   b. Restore the object from the latest saved version.
   c. Allocate the object so no one else can access it.
   d. Restore the needed journal receivers if they are not online. Journal receivers do not need to be restored in a particular sequence. The system establishes the receiver chains correctly when they are restored.
   e. » Use the APYJRNCHG or APYJRNCHGX command to apply the changes to the object.«
   f. Deallocate the object.
5. If an object could not be synchronized, use the information in the history log and in the journal to determine why the object could not be synchronized and how to proceed with recovery. For example, you may need to use the DFU or a user-written program to bring a database file to a usable condition.
6. Determine which applications or programs were active, and determine where to restart the applications from the information in the history log and in the journal.

If a journaled access path is in use during an abnormal system end, that access path does not appear on the Edit Rebuild Access Path display.

If the maintenance for the access path is immediate or delayed, the system automatically recovers the access path during IPL or vary on of an independent disk pool. A status message is displayed for each access path whose maintenance is immediate or delayed as it is being recovered during an IPL or vary on of an independent disk pool. The system places message CPF3123 in the system history log for each access path that is recovered through the journal during the IPL or vary on of an independent disk pool. This message appears for access paths that are explicitly journaled and for access paths that are protected by SMAPP.

Recover a damaged journal receiver

If a journal receiver becomes damaged, the system sends message CPF8136 or message CPF8137 to the system operator and the job log.

If a journal receiver becomes damaged, there are two ways you can recover from it:

- Recover from a damaged receiver manually
- Recover from a damaged receiver with the Work with Journal (WRKJRN) command. It is recommended that you use the WRKJRN command.

Recover from a damaged receiver manually

1. If the damaged receiver is currently attached to a journal, swap the journal receiver to attach a new receiver and detach the damaged receiver.
2. If the journal receiver is not currently attached to a journal, delete the journal receiver and restore a previously saved copy.
3. If the journal receiver was never attached to a journal, delete the receiver and create it again or restore it.

If the journal receiver is partially damaged, all journal entries except those in the damaged portion of the journal receiver can be viewed using the Display Journal (DSPJRN) command. Using this list, you can determine what you need to do to recover your objects. Applying or removing journal changes cannot be done with a partially damaged journal receiver.

Recover from a damaged receiver with the WRKJRN command

To use the Work with Journals display to recover damaged journal receivers, use Option 7 (Recover damaged journal receivers). Option 7 checks to determine which journal receivers that are associated with the specified journal are damaged. If none are damaged, a message appears.

If there are damaged journal receivers associated with the specified journal, the Recover Damaged Journal Receivers display appears and lists those receivers.

The status fields initially show a value of DAMAGED. After recovery has been successfully completed, the status shows a value of RECOVERED (receiver recovered).

To view the online help, type WRKJRN at a command line, and press F1. The online help also contains a description of the journal menus.

Recovery for a damaged journal receiver guides you through the following steps:

1. If the attached receiver is damaged, you must run a Change Journal (CHGJRN) command to attach a new receiver.

   Indicate that you want to create a new receiver. The system presents the Create Journal Receiver (CRTJRNRCV) command prompt for receiver name and attributes. After you create the new receiver, the system shows the CHGJRN command prompt.

   If the attached receiver is not damaged, the preceding step is omitted.
2. The damaged journal receiver is deleted.
3. A prompt for the restore of the damaged journal receiver is shown. Any of the values on the prompt can be changed except the receiver name. Save information in the prompt is provided by the system.

Recover a damaged journal

If a journal becomes damaged, the system sends message CPF8135 to the system operator and to the job log.

Use the following steps to recover a damaged journal:

1. End journaling for all access paths associated with the journal by using the End Journal Access Path (ENDJRNAP) command.
2. End journaling for all physical files associated with the journal by using the End Journal Physical File (ENDJRNPF) command.
3. End journaling for all integrated file system objects by using the End Journal (ENDJRN) command.
4. End journaling for all other object types by using the End Journal Object (ENDJRNOBJ) command.
5. Delete the damaged journal by using the Delete Journal DLTJRN command.
6. Create a journal receiver (CRTJRNRCV command) and create a journal with the same name and in the same library as the damaged journal (CRTJRN command), or restore the journal from a previously saved version.
7. Start journaling the physical files that were journaled by using the Start Journal Physical File (STRJRNPF) command.
8. Start journaling the access paths that were journaled by using the Start Journal Access Path (STRJRNAP) command.
9. Start journaling integrated file system objects with the Start Journal (STRJRN) command.
10. Start journaling other object types with the Start Journal Object (STRJRNOBJ) command.

Note: You can also restore your journaling environment by deleting and restoring all the objects that were being journaled. Objects that were journaled at the time of their save automatically begin journaling at restore time if the journal is online.

11. Save the journaled objects to allow for later recovery.

12. Associate the old journal receivers with the new journal. Do the following:
    a. Type `WRKJRN` and press the Enter key.
    b. On the prompt display, enter the name of the journal.
    c. From the Work with Journal display, select option 9 (Associate receivers).
    d. Press F12 to cancel the display.
    e. Type `WRKJRNA JRN(`*`library-name/journal-name`*`)` and press the Enter key.
    f. From the Work with Journal Attributes display, press F15 to display the receiver directory.

Each time a journal is restored, a new receiver chain is started because the last journal receiver on the chain that existed prior to the restore process did not have the newly created receivers as its next receivers.

Note: If the damaged journal had any remote journals associated with it, use the Add Remote Journal (QjoAddRemoteJournal) API or ADDRMTJRN command to reassociate those remote journals. See Add remote journals for more information.

You can also use the WRKJRN command to recover a damaged journal. However, it is recommended that you use the WRKJRN command to recover a damaged journal if you are only journaling physical files and access paths to this journal.

Associate receivers with journals: Use Option 9 on the Work with Journals display if the journal was restored or created again. The system associates all applicable receivers with the restored or recreated journal so that a restore of these receivers is not necessary.

A journal receiver is associated with a journal if the journal receiver appears in the journal receiver directory. A receiver that was previously attached to a journal but is not currently associated with a journal cannot be used with the journal commands, such as:

- Display Journal (DSPJRN)
- Receive Journal Entry (RCVJRNE)
- Retrieve Journal Entry (RTVJRNE)
- Retrieve Journal Entries (QjoRetrieveJournalEntries) API
- Apply Journaled Changes (APYJRNCHG)
  »
- Apply Journaled Changes Extend (APYJRNCHGX)
  «
- Remove Journaled Changes (RMVJRNCHG)

Recover a damaged journal with the WRKJRN command: The Work with Journal (WRKJRN) command performs all the steps that are described below except for saving the physical files and logical files. The WRKJRN command associates the receivers with the recovered journals without you having to delete and restore the receivers. However, the command only recovers access paths and database files. If you journal other object types you cannot use WRKJRN to recover those object types.

Option 6 on the Work with Journals display verifies that the journal is damaged before proceeding with recovery. If the journal is not damaged, an information message appears.

For a description of the Work with Journals display, see the WRKJRN command in the online command help. To view the help, type WRKJRN on a command line, and press F1.

Recovery for a damaged journal guides you through the following steps:

1. The system attempts to determine which files are currently being journaled to the indicated journal. If the system cannot successfully build this list, a message appears before the recovery operation begins.
2. Journaling, for all access paths that are currently being journaled to the specified journal, is ended.
3. Journaling, for all files that are currently being journaled to the specified journal, is ended .
4. The system deletes the journal.
5. The system presents the Recover Damaged Journal display, which asks you whether to restore or create the journal:
   a. If the journal will be restored, the system prompts for the values that are needed for the restore operation.
   b. If the journal will be created, the system prompts for the receiver names and attributes with the CRTJRNRCV command prompt. The system prompts for values needed to create the journal with the CRTJRN command prompt, with known values that are shown.
6. The list of files for which journaling is to be started again is shown. When you press the Enter key, journaling is started for all files that are listed.
7. The list of files that contains access paths for which journaling is to be started again appears. When you press the Enter key, journaling for the access paths is started for the files that are listed.
8. The system associates all applicable receivers with the re-created or restored journal so that a restore of these receivers is not necessary.

   A journal receiver is associated with a journal if the journal receiver appears in the journal receiver directory. A receiver that was previously attached to a journal, but is not currently associated with a journal, cannot be used with the journal commands such as Display Journal (DSPJRN), Apply Journaled Changes (APYJRNCHG), Apply Journaled Changes Extend (APYJRNCHGX) and Remove Journaled Changes (RMVJRNCHG).

As the recovery of a damaged journal proceeds, the Display Journal Recovery Status display appears. The information about this display is updated as the operation progresses to indicate which steps have been completed, which steps have been bypassed, and which step will be run next. Whenever a user action is required, the status display is replaced by the appropriate prompt display.

The status field indicates the following operation status:
- PENDING. The step has not been started.
- NEXT. The step will be performed next (after the Enter key is pressed).
- BYPASSED. The step was not performed. (It was not necessary).
- COMPLETE. The step has been performed.

The first display you usually see after the first status display is the Recover Damaged Journal display. Use this display to choose whether the journal is to be created or restored.

When the last step of the recovery process is complete, a message appears indicating that all files for which journaling was started must be saved to establish a new recovery point.

If the damaged journal had any remote journals associated with it, use the Add Remote Journal (QjoAddRemoteJournal) API or ADDRMTJRN command to reassociate those remote journals. See Add remote journals for information about adding remote journals.

Recover journaled objects

One of the primary advantages of journaling is its ability to return a journaled object to its current state since the last save. You can recover from many types of damage to journaled objects by using journaled changes. For example, an object is damaged and becomes unusable, an error in an application program caused records to be improperly updated, or incorrect data was used to update an object. In each of these instances, only restoring a saved version of the object can result in the loss of a significant amount of data.

You can use partial receivers to apply or remove changes from an object. If you attempt to restore a saved receiver while a more current version of the receiver is on the system, an escape message is sent to prevent you from restoring the receiver. The system makes sure that the most complete version is preserved.

You can use a partial receiver as the first receiver in the receiver chain for a RMVJRNCHG command only if you specify a sequence number for the FROMENT or FROMENTLRG parameter.

If you use the Apply Journaled Changes (APYJRNCHG) or Apply Journaled Changes Extend (APYJRNCHGX) command to apply journaled changes, significantly less data may be lost. You can use the Remove Journaled Changes (RMVJRNCHG) command to recover from improperly updated records or incorrect data if before-images have been journaled. This command removes (or backs out) changes that were made to an object.

Use the APYJRNCHG command to apply changes to these object types:

- Database file
- Integrated file system object
- Data area

Use the APYJRNCHGX command to apply changes to database files.

Use the RMVJRNCHG command to remove changes that were made to these object types:

- Database file
- Data area

To recover an object by applying or removing journaled changes, the object must be currently journaled. The journal entries must have the same journal identifier (JID) as the object. To ensure the journal identifiers are the same, save the object immediately after journaling is started for the object.

To apply or remove journaled changes to or from a restored copy of the object, you must have already saved the object while it was being journaled. Why you must save objects after you start journaling has more information about saving journaled objects and about JIDs.

If you need to recover objects that were journaled to a journal that you deleted, restore the journal from a saved copy or create a new journal with the same name in the same library. Then restore the object and all the needed receivers before applying or removing journaled changes with that journal. You can use an option on the Work with Journals display to reassociate any journal receivers that are still on the system. To use the Work with Journals display, use the Work with Journals (WRKJRN) command.

Some types of entries in the journal receiver cause the apply or remove process to possibly stop. These entries are written by events that the system cannot reconstruct. Certain illogical conditions, such as a duplicate key in a database file defined as unique, can also cause processing to end.

Use the Object Error Option (OBJERROPT) of the APYJRNCHG, APYJRNCHGX, or RMVJRNCHG commands to determine how the system responds to an error. If you select OBJERROPT(*CONTINUE) and an error occurs, processing of journal entries stops only for the object associated with that error. Processing continues for the other objects. The system sends a diagnostic message indicating that the processing of journaled changes for that object was not successful. The system also places an indication that processing ended early for the specific object in any output file record. If you select OBJERROPT(*END), processing ends for all objects when an error occurs.

Using save-while-active to save your journaled objects can help you recover your objects more quickly when you need to apply or remove journaled changes specifying FROMENT(*LASTSAVE) or FROMENTLRG(*LASTSAVE). When you use the save-while-active function to save your journaled objects, the system saves and then restores information that indicates which starting journal sequence number is needed for the apply or remove operation. When this information is available for all objects to which you are applying or removing journaled changes, the system does not need to scan the journal receivers to determine this starting point. Scanning journal receiver data to find the starting points can be time consuming.

Also, using save-while-active when saving your objects allows you to restore a version of your object which was not from the last save and to still specify FROMENT(*LASTSAVE) or FROMENTLRG(*LASTSAVE) on the apply or remove command and successfully apply or remove changes.

Actions of applying or removing journaled changes by journal code shows how operations that apply and remove journaled changes handle journal entry types. It shows which entry types cause processing to end for an object and what processing is done when the entry is applied or removed. «

The following topics provide information about how to apply and remove journaled changes.

- Apply journaled changes
- Remove journaled changes
  »
- Use the apply and remove journaled changes output file
  «
- Journaled changes with trigger programs
- Journaled changes with referential constraints
- Actions of applying or removing journaled changes by journal code
- When the system ends applying or removing journaled changes
- Example: Apply journaled changes
- Example: Remove journaled changes
  »
- Example: Recover objects with partial transactions
  «

Note: Read the Code example disclaimer for important legal information.

Apply journaled changes: » If an object becomes damaged or is not usable you can recover the object using the Apply Journaled Changes (APYJRNCHG) or Apply Journaled Changes Extend (APYJRNCHGX) command. If you restore an object that was saved with partial transactions, then you must apply journaled changes to that object before it is usable.

Difference between APYJRNCHG and APYJRNCHGX

The difference between APYJRNCHG and APYJRNCHGX is that with APYJRNCHGX you can only specify database files and *ALL files in a library. However, the APYJRNCHGX command can apply journal entries resulting from the following SQL statements:

- CREATE INDEX
- CREATE TABLE
- CREATE VIEW

Object level changes

When you use APYJRNCHG, you can apply most database object level changes with the exception of create database file operations (D CT$^{(TM)}$). With APYJRNCHGX you can apply the same database object level changes as APYJRNCHG and create database file operations. Database object level changes can occur from many CL commands such as Change Physical File (CHGPF), Remove Member (RMVM), or SQL statements such as ALTER TABLE.

The Apply Journaled Changes (APYJRNCHG) and Apply Journaled Changes Extend (APYJRNCHGX) command descriptions have more detailed information about object level changes.

Applying journaled changes to all objects

You can apply journaled changes to all objects that are journaled to the journal by specifying OBJ(*ALLJRNOBJ) on the APYJRNCHG command.

Applying journaled changes and commitment control

You can ensure that commitment transaction boundaries are honored during the apply journaled changes operations by using the commit boundary (CMTBDY) parameter. The default value for the CMTBDY parameter is *YES. If the system encounters a journal entry that causes the apply or remove process to stop for the object, the commitment boundary might not be honored.

Error handling

When the system encounters a journal entry it cannot process, it ends apply processing either for that specific object or for the entire apply operation. You can specify how the system behaves when it encounters a journal entry it cannot process with the Object Error Option (OBJERROPT) parameter on the APYJRNCHG or APYJRNCHGX command. If you specify OBJERROPT(*CONTINUE), the system ends apply processing for the specific object that has an error, but it continues apply processing for the other objects in the apply operation. If you specify OBJERROPT(*END), the system ends processing for the entire apply operation. The OBJERROPT parameter is also available for the Remove Journaled Changes (RMVJRNCHG) command. Actions of applying or removing journaled changes by journal code shows which entry types cause processing to end for an object.

Before you start applying changes

You must first reestablish the object to a condition that you know is undamaged.

- To reestablish the object, restore the last saved copy of the object. The object must have been saved while it was being journaled.
- If you saved a database physical file by using the Copy File (CPYF) command, use the CPYF command to restore the member by overlaying the contents of the existing object with the old values.
- If the member of a database physical file was just initialized, initialize the member again using the Initialize Physical File Member (INZPFM) command or a user-created application program.
- If a member of a database physical file was just reorganized, reorganize the member again using the Reorganize Physical File Member (RGZPFM) command.

You must restore the needed journal receivers if any of the following are true:

- If the journal receivers were deleted since the object was last staved (or some other point).
- If the journal receivers were saved with their storage freed.

When you apply journaled changes to an object, the object cannot be in use by anyone else.

Starting and stopping points for applying journaled changes

When the condition of the object has been established, use the APYJRNCHG or APYJRNCHGX command to apply the changes that are recorded in the journal to the object.

The system applies the changes to the object in the same order as they were originally made. You must plan where you want to start and stop applying changes. Use the Display Journal (DSPJRN) command to identify the desired starting and ending points. If you use a control language (CL) program for your recovery procedures, use the following:

- Receive Journal Entry (RCVJRNE) command to receive journal entries as they are written to the journal receiver.
- Retrieve Journal Entry (RTVJRNE) command to retrieve a journal entry and place it in program variables.

You can also use the QjoRetrieveJournalEntries API to retrieve the information into a High Level Language (HLL) program.

Start the apply

On the APYJRNCHG or APYJRNCHGX command, specify the first journal entry to be applied to the object. This entry can be selected from any of the following points:

- After the last save of the object
- From the first journal entry
- From an identified sequence number that corresponds to a date and time stamp
- From an identified sequence number that corresponds to the start or end of a particular job's use of the object provided that you did not specify one of the following:
  - OMTJRNE(*OPNCLO) when starting journaling or changing the journaling attributes for object
  - OMTJRNE(*OPNCLOSYN) when starting journaling or changing the journaling attributes for a directory or stream file
  - RCVSIZOPT(*MINFIXLEN) for the journal at any time while the object was journaled
  - A FIXLENDTA option that omitted the job name
- A specific sequence number.

Note: If an object was restored with partial transactions, then you must specify FROMENT(*LASTSAVE) or FROMENTLRG (*LASTSAVE).

End the apply

You can stop applying the journal entries at the following:

- The end of the data in the last journal receiver in the receiver range
- A particular entry in the journal
- A date and time stamp
- A commitment boundary
- The start or end of a particular job's use of the data in the object provided you did not specify the following:
  - OMTJRNE(*OPNCLO) when starting journaling or changing the journaling attributes for the object
  - OMTJRNE(*OPNCLOSYN) when starting journaling or changing the journaling attributes for a directory or stream file
  - RCVSIZOPT(*MINFIXLEN) for the journal at any time while the object was journaled
  - A FIXLENDTA option that omitted the job name
- The journal entry that indicates when the object was last restored

- A specific sequence number

»

The apply and remove journaled changes output file

It is highly recommended that you use the apply and remove journaled changes output file when you apply journaled changes. The output file contains a record for each object that the apply operation processes. It contains a record for each object created and each object deleted during the apply. This output file is especially useful when the apply ends early. It is much easier to query the output file for the status of each object rather then searching through the job log messages. Also the messages are limited to 512 while the output file is not limited.

Considerations for applying changes

Considerations for applying changes are as follows:«

- When you apply journaled changes to integrated file system objects, you need to be aware of integrated file system considerations.
- If you are journaling only access paths or database physical files, another way to apply journaled changes is to Apply journaled changes with the WRKJRN command following the prompts.

Integrated file system considerations for applying journaled changes: If there is a create entry or delete entry in the range of journal entries to which you are applying journaled changes, changes to a directory can cause the creation or deletion of an object.

If you are journaling a directory using the **Journal new files and folders** (INHERIT(*YES)) option and an object is created into that directory, the system automatically starts journaling that new object and deposits associated create and start journal object journal entries. The apply of these create and start journal entries during the apply operation on the directory then creates the objects and starts journaling for them during the apply operation. For any subsequent journaled entries for that object, the apply operation applies any entries that it encounters for that object as well. Similarly, if an entry is encountered which deletes (unlinks) an integrated file system object, that object is actually deleted as part of the apply operation.

Additionally, the apply operation will start journaling for any integrated file system journal entry that adds a link to the journaled directory, such as moving a nonjournaled object into the journaled directory, or adding a new hard link to a nonjournaled object into this journaled directory. However, no entries will be applied to these objects since the state of those objects is not fully know during the apply.

As objects are created, they are included in the maximum number of objects which can be applied as part of one Apply Journaled Changes (APYJRNCHG) request.

»

Error handling considerations

When you apply journaled changes, you can use the Object Error Option (OBJERROPT) of the APYJRNCHG command to specify how the system responds to errors. If you specify *CONTINUE, the system stops applying changes to the object that encounters an error, but continues the apply operation for the remaining objects.

For integrated file system objects, the system processes errors for directory-level operations separately from object-level operations. For example, you perform an apply journaled changes operation for a directory and a stream file in that directory. During the apply operation, an error occurs for the stream file and the apply process ends for that stream file. You might expect some operations that are associated with that stream file, such as remove link, to end also. But since remove link is a directory level operation, the remove link operation still occurs, even though the apply operation ended for that stream file.

Therefore even though object-level operations for an object might end, directory-level operations that are associated with that object still occur.

«

Commitment control considerations

Many journaled integrated file system operations use system initiated commitment control for the duration of the operation. These operations are not considered completed successfully unless the commitment control cycle is committed. Commitment control, here, refers to commitment control that the system initiates. Integrated file system operations cannot be included in a user initiated commitment control cycle.

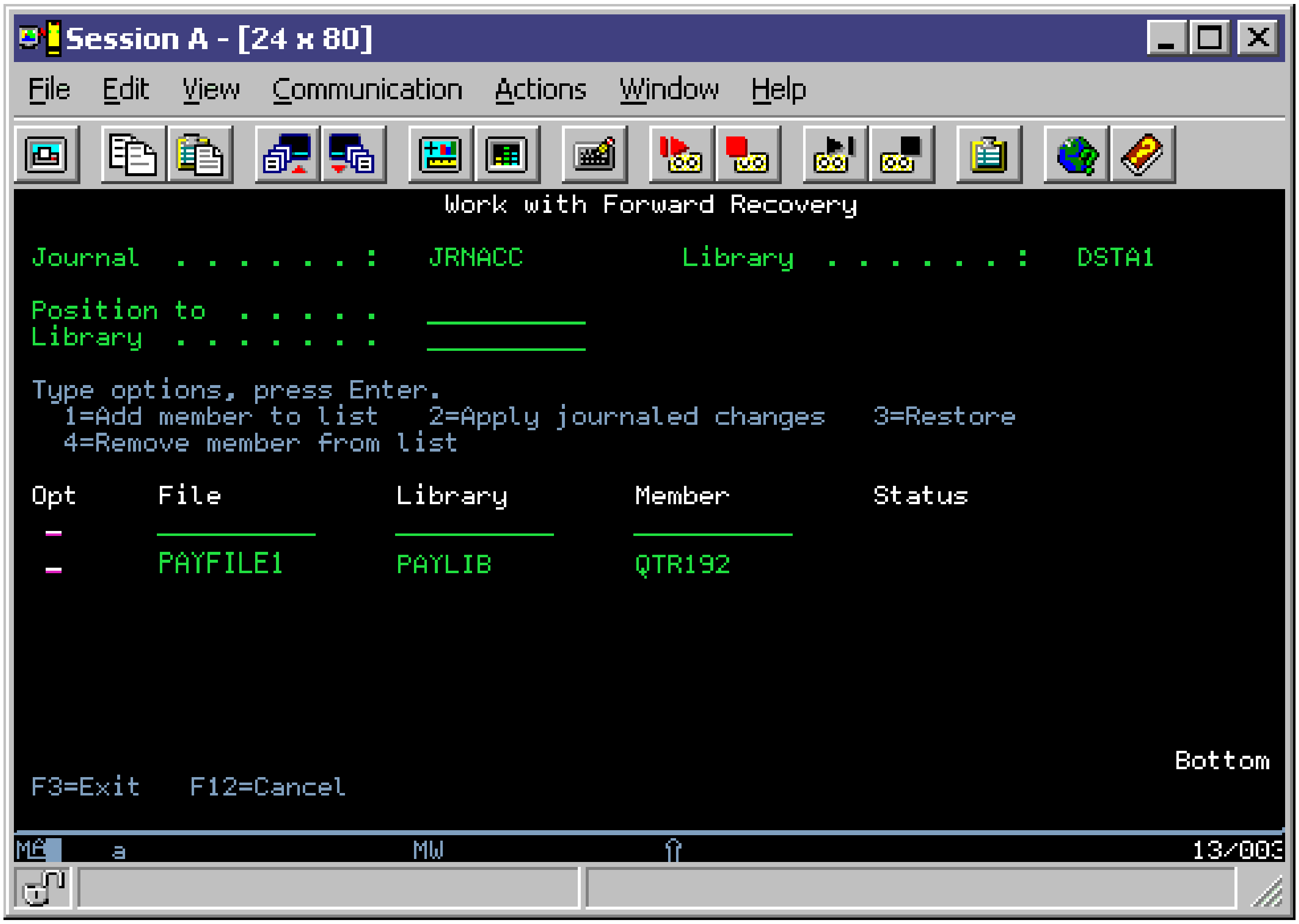
» For integrated file system journal entries that are part of a commitment control cycle, do not apply individual entries from within the cycle without applying the entire commit cycle. Using the Commit Boundary (CMTBDY(*YES)) parameter on the APYJRNCHG command can help enforce this. If you do not use this option and choose a specific starting point, start from the Start of commit cycle (C SC) entry for that cycle. Likewise, if you choose to end applying a journaled change at a specific point, end on the Commit (C CM) or Rollback (C RB) entry for that cycle. «

See Actions of applying or removing journaled changes by journal code for what operations are applied for integrated file system related journal entries.

Apply journaled changes with the WRKJRN command: The Work With Journal (WRKJRN) command only works if you are journaling only access paths or database files. To apply journaled changes with the WRKJRN command select Option 2 (Work with Forward Recovery). The Work with Forward Recovery display contains a status field for each file member. For a description of the journal options, see the online information for the WRKJRN command by pressing F1. The status field for each member indicates the following:

- NOT FOUND
- DAMAGED
- NOT SYNCHRONIZED
- RESTORE COMPLETE
- RECOVERED
- NOT JOURNALED
- DIFFERENT JOURNAL
- Blank

The Work with Forward Recovery display looks like the following figure:



```
                          Work with Forward Recovery

 Journal . . . . . . . :   JRNACC          Library . . . . . . :   DSTA1

 Position to . . . . .     __________
 Library . . . . . . .     __________

 Type options, press Enter.
   1=Add member to list    2=Apply journaled changes   3=Restore
   4=Remove member from list

 Opt     File              Library           Member            Status
  _      __________        __________        __________
  _      PAYFILE1          PAYLIB            QTR192




                                                                        Bottom
 F3=Exit   F12=Cancel
```

Tasks with the Work With Forward Recovery display

You can use the Work With Forward Recovery display to perform the following tasks:

Add member to list
To add a member to the list on the display use Option 1 (Add member to list) to add a member to the list. Do this if you want to restore those members.

Apply journaled changes
To apply journaled changes to a member use Option 2 (Apply journaled changes). This option applies journaled changes and changes the status to RECOVERED (if the apply operation was successful). If the apply operation was not successful, messages appear indicating why, and the status remains the same. If any required receivers are missing or damaged while running the APYJRNCHG command, the system displays prompts for the restore procedures for the missing or damaged receivers.

If any of the members in the list have a status of DAMAGED when use option 2, the system prompts you with the command necessary to recover the file member. For files that are damaged, recovery involves the restore of the last save that is followed by the Apply Journaled Changes (APYJRNCHG) command. The system guides you through recovery as follows:

1. The system identifies all the logical files dependent on the specified damaged file. The Dependent Logical Files display appears identifying these files.
2. The dependent logical files are deleted.
3. The system deletes the files to be recovered (or restored).

4. The system displays prompts for the restore of files to be recovered. After all restores are completed successfully, the files to be recovered are allocated exclusively to prevent any other processing. This allocation is maintained until the recovery procedures are complete.
5. The system displays prompts for the restores of the dependent logical files.
6. » An APYJRNCHG command is prompted.«
7. If the APYJRNCHG command encounters a required journal receiver that is not online, the system prompts for the restore of the required receiver and again starts the APYJRNCHG command.

When the recovery process is complete, the status field for the member indicates RECOVERED (if the operation was successful). If the operation failed, the status field remains unchanged, and messages appear indicating why the operation failed.

Restore members with status of NOT FOUND
If any members have a status of NOT FOUND use Option 3 (Restore). This option prompts you for the files to restore. Members that are restored successfully have a status of RESTORE COMPLETE. Members that are not restored keep their old status. A message is sent indicating that the restore did not complete successfully. All members that are restored are included in the list of members to recover.

Note: The last save information is provided for the restore operation. If either of the following are true, the you must use the RSTOBJ command instead of Option 3 (Restore):

- The device provided is tape, diskette, or optical and you choose to restore from a save file (*SAVF).
- The device provided is a save file (*SAVF) and you choose to restore from tape, diskette, or optical media.

Remove member from list
To remove a member from the list, use Option 4 (Remove member from list). Option 4 removes file members from the list of members to be recovered.

Remove journaled changes: Depending on the type of damage to the journaled object and the amount of activity since the object was last saved, removing changes from the object can be easier than applying changes to the object. Use the Remove Journaled Changes (RMVJRNCHG) command to remove changes from an object if you are journaling before-images.

The RMVJRNCHG command removes changes in reverse chronological order, starting with the most recent change.

On the RMVJRNCHG command, you identify the first journal entry to be removed from the object. This entry can be from:

- The last journal entry that is contained within the range of journal receivers specified
- The entry that corresponds to the last save of the object
- An identified sequence number

You can control the changes that are removed from the object. For example, assume that an application updated data incorrectly for a period of time. In this case, you can remove the changes from the object until that application first opened the object.

You can stop removing journaled changes at:

- » The start of the commit cycle for a transaction. «

- The end of data in the journal receivers. This corresponds to the first journal entry that was recorded on the range of journal receivers that are specified.
- An identified sequence number that corresponds to a particular entry in the journal.
- The start of a particular job's use of the object. You can only specify this if you did not specify any the following:
  - To exclude open and close journal entries (OMTJRNE(*OPNCLO)) when starting journaling for the file
  - To minimize fixed-length entries RCVSIZOPT(*MINFIXLEN) for the journal at any time while the object was journaled.
  - To omit a FIXLENDTA option that includes the job name.

You can ensure that commitment transaction boundaries are honored on the remove journaled changes operations by using the CMTBDY parameter on these commands.

If the system encounters a journal entry that causes the apply or remove process to stop, the commitment boundary may not be honored.

Error handling

When the system encounters a journal entry it cannot process, it ends remove processing either for that specific object or for the entire remove operation. You can specify how the system behaves when it encounters a journal entry it cannot process with the Object Error Option (OBJERROPT) on the Remove Journaled Changes (RMVJRNCHG) command. If you specify OBJERROPT(*CONTINUE), the system ends remove processing for the specific object, but it continues remove processing for the other objects in the remove operation. If you specify OBJERROPT(*END), the system ends processing for the entire remove operation. Actions of applying or removing journaled changes by journal code shows which entry types cause processing to end for an object .

Starting and ending points

Use the Display Journal (DSPJRN) command to identify the required starting and ending points. If you use a control language (CL) program for your recovery procedures, use the following:

- Receive Journal Entry (RCVJRNE) command to receive journal entries as they are written to the journal receiver.
- Retrieve Journal Entry (RTVJRNE) command to retrieve a journal entry and place it in program variables.

You can also use the Retrieve Journal Entries (QjoRetrieveJournalEntries) API to retrieve the information into a High Level Language (HLL) program.

Another way to remove journaled changes is to Remove journaled changes with the WRKJRN command and follow the command prompts.

The apply and remove journaled changes output file

It is highly recommended that you use the apply and remove journaled changes output file when you remove journaled changes. The output file contains a record for each object that the remove operation processes. It contains a record for each object created and each object deleted during the remove. This output file is especially useful when the remove ends early. It is much easier to query the output file for the status of each object rather then searching through the job log messages. Also the messages are limited to 512 while the output file is not limited.

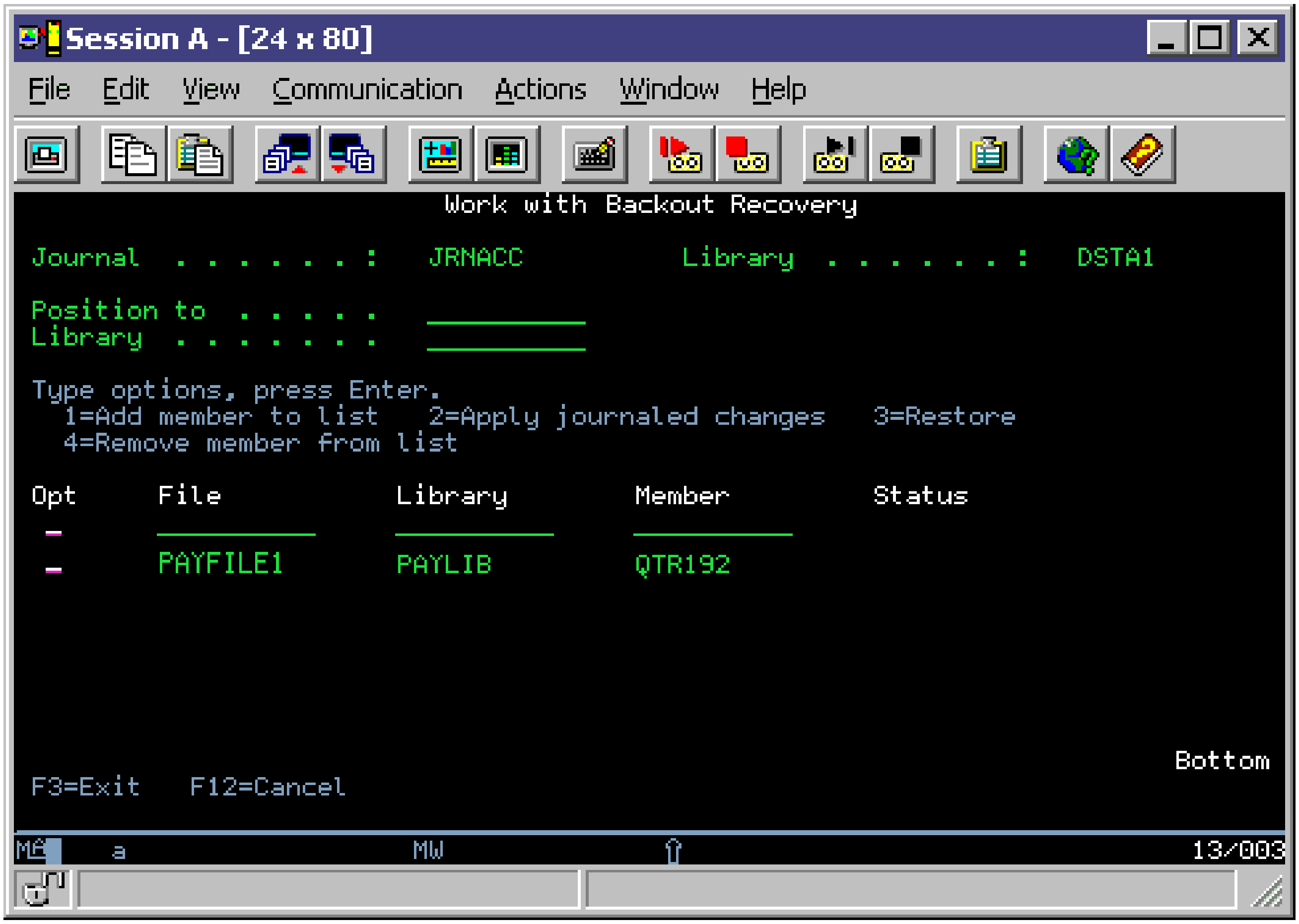
«

Remove journaled changes with the WRKJRN command: To remove journaled changes with the Work With Journal (WRKJRN) command select Option 3 (Backout recovery). The Work with Backout Recovery display shows list of the file members that are being journaled.

The Work with Backout Recovery display is useful because the system guides you through the process. However, it works if you are journaling access paths or database files only.

The same options on the Work with Forward Recovery display are available on the Work with Backout Recovery display. However, the option to restore the file is not valid for backout recovery. The status field that is shown on the Work with Backout Recovery display is either blank or it indicates the same status as for forward recovery, except for RESTORE COMPLETE.

For a description of the journal options, see the online information for the WRKJRN command by pressing F1.



```
                          Work with Backout Recovery

Journal  . . . . . . :   JRNACC          Library  . . . . . . :   DSTA1

Position to  . . . . .   __________
Library  . . . . . . .   __________

Type options, press Enter.
  1=Add member to list   2=Apply journaled changes   3=Restore
  4=Remove member from list

Opt     File           Library        Member         Status
 _      __________     __________     __________
 _      PAYFILE1       PAYLIB         QTR192




                                                                        Bottom
F3=Exit   F12=Cancel
MA     a                  MW              ⇧                          13/003
```

Tasks with the Work With Backout Recovery display

You can use the Work With Backout Recovery display to perform the following tasks:

Add member to list
To add a member to the list select Option 1 (Add member to list).

Remove journaled changes
To remove journaled changes, select Option 2 (Remove journaled changes). Option 2 shows the

Remove Journaled Changes (RMVJRNCHG) command prompt, removes the journaled changes, and changes the status to RECOVERED (if the operation was successful). If any required journal receivers are missing or damaged while the RMVJRNCHG command is running, the system displays prompts for the necessary restore procedures for the missing or damaged receivers. If the remove operation was not successful, messages appear indicating why the status remains the same.

If any members in the list have a status of NOT FOUND or DAMAGED when on the Work with Backout Recovery display, the operation is not allowed. These members must be recovered in a forward fashion after they have been restored. Forward recovery of specific files must be used for this type of recovery.

Remove member from list
Use Option 4 (Remove member from list) to remove file members from the list.

Use the apply and remove journaled changes output file: » Use the apply and remove journaled changes output (QAJRNCHG) file to make a record of all the activity that occurs when you perform an apply or remove journaled changes operation.

When you specify to create the output file, the system uses the QAJRNCHG output file in the QSYS library with the format name QJOAPYRM as a model.

The words in parenthesis in the Field column indicate the column heading used in the output file.

See the following commands for the all of the parameters used with this output file:

- Apply Journaled Changes (APYJRNCHG)
- Apply Journaled Changes Extend (APYJRNCHGX)
- Remove Journaled Changes (RMVJRNCHG)

The following table describes the fields that the output file creates.

Relative offset	Field	Format	Description
Fields defining the header information			
1	Command (QJOCMD)	Char (10)	Indicates if APYJRNCHG, APYJRNCHGX, or RMVJRNCHG was used.
11	Detail option (QJODET)	Char (1)	Specifies the level of detail that was selected for this output file: **A** = DETAIL(*ALL) The file contains information about the command and an entry for each object that was applied to, whether it existed when the apply command started or it was created during the apply. **E** = DETAIL(*ERR) The file contains information about the command and an entry only for each object that was not successfully applied to. If the apply ends early for an object an entry is included for it.
12	System (QJOSYS)	Char (8)	The name of the system where the apply or remove journaled changes operation was performed.
20	Release (QJOSRL)	Char (6)	The release of OS/400(R) that the system performing the apply or remove operation uses.

Relative offset	Field	Format	Description
26	Journal name (QJOJRN)	Char (10)	The name of the journal.
36	Library name (QJOJLB)	Char (10)	The name of the library for the journal.
46	ASP device (QJOASP)	Char (10)	The name of the auxiliary storage pool (ASP) device for the library.
56	Commit boundary (QJOCMT)	Char (1)	Indicates if a commit boundary was used in the apply or remove operation. **Y** = CMTBDY(*YES) was specified **N** = CMTBDY(*NO) was specified
57	Reserved (QJORS1)	Char (30)	Reserved field
Results summary fields			
87	Number of objects (QJONOB)	Char (10)	Total number of objects processed during the apply or remove operation.
97	Total entries (QJONEN)	Char (20)	Total number of entries processed during the apply or remove operation.
117	Last entry (QJOLST)	Char (20)	Last entry examined in the apply or remove operation.
137	End partial LUW (QJOLUW)	Char (1)	At least one transaction was omitted because CMTBYD(*YES) was specified and the ending sequence number was not at a commit boundary. **Y** = Yes **N** = No
138	Reserved (QJORS2)	Char (20)	Reserved field
Object apply or remove information			
158	Object deleted (QJOOSD)	Char (1)	Indicates if the object was deleted during the apply or remove operation. **Y** = Yes **N** = No
159	Object created (QJOOSC)	Char (1)	Indicates if the object was created during the apply or remove operation. **Y** = Yes **N** = No
160	Early end (QJOOSE)	Char (1)	Indicates if the apply or remove operation ended early for this object. **Y** = Yes **N** = No
161	Change not made (QJOOSU)	Char (1)	Indicates that a change was found for this object after an early end to the apply operation. **Y** = Yes **N** = No

Relative offset	Field	Format	Description
162	End reason code (QJORCD)	Hex (1)	Reason code for early end. See message MCH4801 for the possible values.
163	End message ID (QJOMID)	Char (7)	Message identifier associated with an early end to the apply operation.
170	Error condition (QJOENO)	Hex (4)	Error condition associated with an early end to the apply operation.
174	Partial transactions remain (QJOPTL)	Char (1)	Changes for partial transactions remain for this object. **Y**= Yes **N**= No
175	Partial transactions removed (QJOPTR)	Char (1)	Indicates whether partial transactions were removed for this object. **Y**= Yes **N**= No
176	Reserved (QJORS3)	Char (20)	Reserved field
196	Starting sequence number (QJOSSN)	Char (20)	Specified starting sequence number for the apply or remove operation.
216	Starting receiver name (QJOSRC)	Char (10)	The name of the first receiver from which entries were applied or removed.
226	Receiver library (QJOSLB)	Char (10)	The library for the starting journal receiver.
236	Ending sequence number (QJOESN)	Char (20)	Specified ending sequence number for the apply or remove operation.
256	Ending receiver name (QJOERC)	Char (10)	The name of the last or ending receiver from which entries were applied or removed.
266	Library name (QJOERL)	Char (10)	The library for the ending journal receiver.
276	First entry applied or removed (QJOASN)	Char (20)	The first entry of the apply or remove operation.
296	Last entry applied or removed (QJOAEN)	Char (20)	The last entry of the apply or remove operation.
316	Number of entries (QJONUM)	Char (20)	The number of journal entries that were applied or removed.
336	Partial transaction starting sequence number (QJOBSN)	Char (20)	Starting sequence number for any partial transactions that were removed. For integrated file system objects, this field is always zero.
356	Partial transaction ending sequence number (QJOBEN)	Char (20)	Ending sequence number for any partial transactions that were removed. For integrated file system objects and data areas, this field is always zero.
376	Number of partial transaction removed (QJOBNM)	Char (20)	Count of number of entries removed for partial transactions. For integrated file system objects and data areas, this number is always zero.

Relative offset	Field	Format	Description
396	No entries applied indicator (QJONAIN)	Char (1)	Indicates why no entries were applied to the object. **1** = The object was created during apply, but did not get journaled or can never be journaled. **2** = The object existed before the apply and was journaled as a result of the apply. However, no entries were applied because it could not be determined that the correct version of the object was on the server at the time of the apply.
397	Reserved (QJORS4)	Char (19)	Reserved field
Object identification information			
416	Object type (QJOOTP)	Char (10)	The type of object.
426	Object name (QJOONM)	Char (10)	The name of the object.
436	Object library (QJOOLB)	Char (10)	The object's library.
446	Member name (QJOOMB)	Char (10)	Member name
456	FID (QJOOFD)	Char (16)	The file identifier of an integrated file system object
472	Path indicator (QJOAPI)	Char (1)	The absolute or relative path indicator. This field uses one of the following values: **0 =** The path contains an absolute path name. The Relative directory FID field is hex zeros. **1 =** The path contains a relative path name. The Relative directory FID field is valid and can be used to form a complete path name. This field only applies to integrated file system objects.
473	Relative directory file ID (QJORPI)	Char (16)	The path contains a relative path name. The Relative directory file ID field is valid and can be used to form a complete path name. This field only applies to integrated file system objects.
489	Path name CCSID (QJOPCC)	Hex (4)	The coded character set identifier (CCSID) for the path name. This field only applies to integrated file system objects.
493	Path name region ID (QJOPRE)	Char (2)	The region or country identifier for national language support. This field only applies to integrated file system objects.
495	Path name language ID (QJOPLN)	Char (3)	The language identifier national language support. This field only applies to integrated file system objects.
498	Reserved (QJORS5)	Char (3)	Reserved field
501	Path name type (QJOPNT)	Hex (4)	THE path name type uses one of the following values: **0 =** The path name is a character string with a one byte delimiter. **2 =** The path name is a character string with a two byte delimiter. This field only applies to integrated file system objects.
505	Path name length (QJOPNL)	Hex (4)	The length of the path name. This field only applies to integrated file system objects.
509	Path name delimiter (QJOPND)	Char (2)	The path name delimiter. This field only applies to integrated file system objects.

Relative offset	Field	Format	Description
511	Reserved (QJORS6)	Char (8)	Reserved field
519	Path name (QJOPNM)	Char (5000)	The path name. The length of this field is variable, depending on the path name. This field only applies to integrated file system objects.

«

Journaled changes with trigger programs: The system does not call trigger programs when it is applying or removing journal entries. If an event occurs that would normally cause a trigger program to run, it is up to you to ensure that the processing performed by the trigger program is recovered correctly.

Normal recovery processing will work correctly if all of the following are true:

- The trigger program only performs processing on object types which can be journaled and applied
- The processed object types are journaled
- Journaled changes are applied to or are removed from all the objects that are affected by the trigger program

If additional work is performed by the trigger program or objects other than object types which can be journaled and applied are updated, you must use user-written programs to recover the work performed by the trigger program.

If you use trigger programs to perform these actions, consider using the Send Journal Entry (QJOSJRNE) API to send journal entries when trigger programs are called. See Send your own journal entries. To help with recovery, you can develop a program to retrieve these entries and perform the same operations.

The output format for journal entries (except the *TYPE1, *TYPE2, and *TYPE3 formats) and the QjoRetrieveJournalEntries API interface include information about whether a journal entry was created because of actions that were performed when a trigger program was called.

Journaled changes with referential constraints: When you apply or remove journaled changes, journal management does not support referential constraints. In the following cases, files may be in CHECK PENDING status after you have applied or removed journaled changes:

- When you restore a file that already exists, the referential constraints for the system copy of the file are used. Some of the journaled changes that you apply may have been valid with the referential constraints that were associated with the saved copy. However, they are not necessarily valid with the current referential constraints. If you have changed the referential constraints on the file, considering doing one of the following before applying or removing journaled changes:
  - Deleting the system copy and then restoring the file
  - Recreating the changes to the referential constraints

  When you apply or remove journaled changes, the system attempts to verify the referential constraints at the end of the command, before returning control to you. This may result in a CHECK PENDING status.
- Some referential constraints cause an action to another file. You may define a constraint so that deleting a record in one file causes a related record to be deleted in another file. Because referential constraints are not enforced when you apply journaled changes, the second delete operation does not happen automatically. However, if you are journaling both files and applying journaled changes to both files, the system applies the journal entry for the second file when it encounters it.

  If one of the files in a referential constraint was not journaled or is not included when you apply or remove journaled changes, the referential constraint will probably be put in CHECK PENDING status.

The output format for journal entries (except the *TYPE1, *TYPE2, and *TYPE3 formats) and the QjoRetrieveJournalEntries API interface include information about whether a journal entry was created because of changes that occurred to a record that was part of a referential constraint.

Actions of applying or removing journaled changes by journal code: The following table shows the actions that are taken by the Apply Journaled Changes (APYJRNCHG), Apply Journaled Changes Extend (APYJRNCHGX), or Remove Journaled Changes (RMVJRNCHG) command by journal code and entry type. If All is specified for the **Entry Type**, it indicates that all entry types for that journal code have the specified actions taken by the APYJRNCHG, APYJRNCHGX, or RMVJRNCHG command.

When the system ends applying or removing journaled changes has detailed information about when an apply or remove journaled changes action ends automatically.

Actions by journal code and entry type

Journal code	Entry type	Operation	APYJRNCHG	APYJRNCHGX	RMVJRNCHG
A	All		Ignores	Ignores	Ignores
B	AA	Change audit attribute	Attribute is changed	Ignores	Ignores
B	AJ	Start of apply	Ends for this object[3]	Ignores	Ignores
B	AT	End of apply	Ends for this object[3]	Ignores	Ignores
B	BD	Integrated file system object deleted	Ignores	Ignores	Ignores
B	B0	Begin create	Ignores	Ignores	Ignores
B	B1	Create summary	Object is created and linked	Ignores	Ignores
B	B2	Link to existing object	Object is linked	Ignores	Ignores
B	B3	Rename, move object	Object is moved or renamed	Ignores	Ignores
B	B4	Remove link (parent directory)	Object link is removed	Ignores	Ignores
B	B5	Remove link (link)	Object link is removed	Ignores	Ignores
B	B6	Bytes cleared, after-image	Object is updated	Ignores	Ignores
B	CS	Integrated file system object closed	Ignores	Ignores	Ignores
B	ET	End journaling for object	Ends for this object[3]	Ignores	Ignores
B	FA	Integrated file system object attribute changed	Attribute is changed	Ignores	Ignores
B	FC	Integrated file system object forced	Ignores	Ignores	Ignores
B	FF	Storage for object freed	Ignores	Ignores	Ignores
B	FR	Integrated file system object restored	Ends for this object[3]	Ignores	Ignores

Journal code	Entry type	Operation	APYJRNCHG	APYJRNCHGX	RMVJRNCHG
B	FS	Integrated file system object saved	Ignores	Ignores	Ignores
B	FW	Start of save	Ignores	Ignores	Ignores
B	JA	Change journaled objects attribute	Journal attribute changed	Ignores	Ignores
B	JT	Start journaling for object	Ignores	Ignores	Ignores
B	OA	Change object authority	Authority is changed	Ignores	Ignores
B	OF	Integrated file system object opened	Ignores	Ignores	Ignores
B	OG	Change primary group	Primary group is changed	Ignores	Ignores
B	OI	Object in use at abnormal end, object is synchronized[1]	Ignores	Ignores	Ignores
B	OI	Object in use at abnormal end, object is not synchronized[1]	Ends for this object[3]	Ignores	Ignores
B	OO	Change Object Owner	Owner is changed	Ignores	Ignores
B	RN	Rename file identifier	File identifier renamed	Ignores	Ignores
B	TR	Integrated file system object truncated	Object is truncated	Ignores	Ignores
B	WA	Write, after-image	Object is updated	Ignores	Ignores
C	All		Ignores	Ignores	Ignores
D	AC	Add RI constraint	Constraint is added	Constraint is added	Ignores
D	CG	Change file	File is changed	File is changed	Ignores
D	CT	Create database file	Ignores	File is created	Ignores
D	DC	Remove RI constraint	Constraint is removed	Constraint is removed	Ignores
D	DD	End of apply	Ends for this object[3]	Ends for this object[3]	Ignores
D	DF	Delete file	Ignores	Ignores	Ignores
D	DG	Start of Apply	Ends for this object[3]	Ends for this object[3]	Ignores
D	DH	File saved	Ignores	Ignores	Ignores
D	DJ	Changed journaled object attribute	Journal attribute changed	Ignores	Ignores
D	DT	Delete file	File is deleted	File is deleted	Ignores
D	DW	Start of save	Ignores	Ignores	Ignores
D	DZ	File restored	Ends for this object[3]	Ends for this object[3]	Ignores
D	EF	End journal for file	Ends for this object[3]	Ends for this object[3]	Ignores
D	FM	File moved	File is moved	File is moved	Ignores

Journal code	Entry type	Operation	APYJRNCHG	APYJRNCHGX	RMVJRNCHG
D	FN	File renamed	File is renamed	File is renamed	Ignores
D	GC	Change constraint	Constraint is changed	Constraint is changed	Ignores
D	GO	Change owner	Owner is changed	Owner is changed	Ignores
D	GT	Grant authority	Authority is granted	Authority is granted	Ignores
D	ID	File in use	Ignores	Ignores	Ignores
D	JF	Start journaling file	Ignores	Ignores	Ignores
D	MA	Member added	List of objects being applied is updated.	List of objects being applied is updated.	Ignores
D	RV	Revoke authority	Authority is revoked	Authority is revoked	Ignores
D	TC	Create trigger	Trigger is created	Trigger is created	Ignores
D	TD	Remove trigger	Trigger is removed	Trigger is removed	Ignores
D	TG	Change trigger	Trigger is changed	Trigger is changed	Ignores
D	TQ	Refresh table	Table is refreshed	Table is refreshed	Ignores
E	EA	Update data area, after image	Data area modified	Ignores	Ignores
E	EB	Update data area, before image	Ignores	Ignores	Data area modified
E	ED	Data area deleted	Ends for this object[3]	Ignores	Ends for this object[3]
E	EG	Start journal for data area	Ignores	Ignores	Ends for this object[3]
E	EH	End journal for data area	Ends for this object[3]	Ignores	Ignores
E	EI	Data area in use, object synchronized[1]	Ignores	Ignores	Ignores
E	EI	Data area in use, object not synchronized[1]	Ends for this object[3]	Ignores	Ends for this object[3]
E	EK	Change journaled objects attribute	Attribute changed	Ignores	Ignores
E	EL	Data area restored	Ends for this object[3]	Ignores	Ends for this object[3]
E	EM	Data area moved	Date area is moved	Ignores	Ignores
E	EN	Data area renamed	Data area is renamed	Ignores	Ignores
E	EQ	Data area changes applied	Ends for this object[3]	Ignores	Ends for this object[3]
E	ES	Data area saved	Ignores	Ignores	Ignores
E	EU	RMVJRNCHG command started	Ends for this object[3]	Ignores	Ends for this object[3]
E	EW	Start of save for data area	Ignores	Ignores	Ignores

Journal code	Entry type	Operation	APYJRNCHG	APYJRNCHGX	RMVJRNCHG
E	EX	Data area changes removed	Ends for this object[3]	Ignores	Ends for this object[3]
E	EY	APYJRNCHG command started	Ends for this object[3]	Ignores	Ends for this object[3]
F	AY	Journaled changes applied	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	CB	Change File member	Member is changed	Member is changed	Ignores
F	CE	Change end of data	Member end of data changed[2]	Member end of data changed[2]	Ends for this object[3]
F	CH	File changed	Ignores	Ignores	Ignores
F	CL	Member closed	Ignores	Ignores	Ignores
F	CR	Member cleared	Member cleared of all records[2]	Member cleared of all records[2]	Ends for this object[3]
F	C1	End Rollback	IF CMTBDY(*NO) is selected, ends for this object. If CMTBDY(*YES) is selected, ignores.	IF CMTBDY(*NO) is selected, ends for this object. If CMTBDY(*YES) is selected, ignores.	IF CMTBDY(*NO) is selected, ends for this object. If CMTBDY(*YES) is selected, ignores.
F	DE	Member deleted record count	Ignores	Ignores	Ignores
F	DM	Delete member	Member is deleted	Member is deleted	Ignores
F	EJ	End journaling	Ends for this object[3]	Ends for this object[3]	Ignores
F	EP	End journaling access paths	Ignores	Ignores	Ignores
F	FD	Member forced to auxiliary storage	Ignores	Ignores	Ignores
F	FI	Internal format information	Ignores	Ignores	Ignores
F	IU	Member in use at abnormal end, object synchronized[1]	Ignores	Ignores	Ignores
F	IU	Member in use at abnormal end, object not synchronized[1]	Ends for this object	Ends for this object	Ends for this object
F	IZ	Member initialized	Initialized records inserted in member	Initialized records inserted in member	Initialized records deleted from member
F	JC	Change journal attribute	Ignores	Ignores	Ignores
F	JM	Start journaling member	Ignores	Ignores	Ends for this object[3]
F	JP	Start journaling access paths	Ignores	Ignores	Ignores
F	MC	Create member	Member is created	Member is created	Ignores
F	MD	Member deleted	Ignores	Ignores	Ends for this object[3]

Journal code	Entry type	Operation	APYJRNCHG	APYJRNCHGX	RMVJRNCHG
F	MF	Member saved with storage freed	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	MM	Member moved	Member is moved	Member is moved	Ignores
F	MN	Member renamed	Member is renamed	Member is renamed	Ignores
F	MO	Member changed	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	MR	Member restored	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	MS	Member saved	Ignores	Ignores	Ignores
F	OP	Member opened	Ignores	Ignores	Ignores
F	PD	Access path deleted	Ignores	Ignores	Ignores
F	PM	Logical owning member of access path moved	Ignores	Ignores	Ignores
F	PN	Logical owning member of access path renamed	Ignores	Ignores	Ignores
F	RC	Journaled changes removed	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	RG	Member reorganized	Ignores	Ignores	Ends for this object[3]
F	RM	Member reorganized	Member is reorganized	Member is reorganized	Ignores
F	SA	Start of APYJRNCHG	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	SR	Start of RMVJRNCHG	Ends for this object[3]	Ends for this object[3]	Ends for this object[3]
F	SS	Start of save active	Ignores	Ignores	Ignores
I	All		Ignores	Ignores	Ignores
J	All (Except SI and SX)		Ignores	Ignores	Ignores
J	SI	Enter JRNSTATE(*STANDBY)	Ends	Ignores	Ignores
J	SX	Exit JRNSTATE(*STANDBY)	Ignores	Ignores	Ends
L	All		Ignores	Ignores	Ignores
M	All		Ignores	Ignores	Ignores
O	All		Ignores	Ignores	Ignores
P	All		Ignores	Ignores	Ignores
Q	All	Data queue functions	Ignores	Ignores	Ignores
R	BR	Before-image updated for rollback operation	Ignores	Ignores	Record updated with before-image
R	DL	Record deleted	Record deleted	Record deleted	Record updated with before-image
R	DR	Record deleted for rollback operation	Record deleted	Record deleted	Record updated

Journal code	Entry type	Operation	APYJRNCHG	APYJRNCHGX	RMVJRNCHG
R	IL	Increment record limit	Ignores	Ignores	Ignores
R	PT	Record written to member	Record written to member	Record written to member	Record deleted from member
R	PX	Record added directly to member	Record added	Record added	Record deleted from member
R	UB	Record updated (before-image)	Ignores	Ignores	Record updated with before-image
R	UP	Record updated (after-image)	Record updated with after-image	Record updated with after-image	Ignores
R	UR	After-image updated for rollback operation	Record updated with after-image	Record updated with after-image	Ignores
S	All		Ignores	Ignores	Ignores
T	All		Ignores	Ignores	Ignores
U	User-specified	User entry	Ignores	Ignores	Ignores

Notes:

[1] The **Flag** field in the journal entry indicates whether the object is synchronized (0 = object was synchronized; 1 = object was not synchronized).

[2] Applying journaled changes stops at this entry if referential constraints that this entry violates are active during the apply operation.

[3] Any changes found for the object that follow this entry are not applied. If any additional changes are found for this object an indication will be returned in the end of apply or remove journal entry, and in any output file generated. If you specify *END for the Object Error Option (OBJERROPT) when you issue the apply or remove journaled changes command, the entire apply or remove operation ends.

«

For more information about the journal codes, entry types, and journal entries, see Journal entry information.

When the system ends applying or removing journaled changes: The system ends applying or removing journaled changes as a result from one of the following items:

- Certain journaled entries
- A format error for a database physical file (such as an undefined entry for that file member)
- A logical error for a database physical file (such as updating a record that has not been inserted or a duplicate key exception)

»

When one of the previous items occur, the apply or remove journaled changes action can end either for the object or for the entire apply or remove operation. You can determine this behavior by using the Object Error Option (OBJERROPT) parameter on the Apply Journaled Changes (APYJRNCHG), Apply Journaled Changes Extend (APYJRNCHGX), or Remove Journaled Changes (RMVJRNCHG) commands.

When OBJERROPT(*END) is specified, for entries that end applying or removing journaled changes, a message identifying the reason for the end is placed in the job log, and the corresponding change is not made to the object. The message contains the sequence number of the journal entry on which the failing condition was detected. When OBJERROPT(*CONTINUE) is specified, message CPD7016 indicates what entry the apply or removed stopped at, and the reason code. This information is also available in the output file if one was generated. To correct the problem do the following:

1. Analyze the error.
2. Make the necessary correction.
3. Start applying or removing journal changes again using the appropriate sequence number.

«

For example, if the entry that causes a RMVJRNCHG command to end is entry code F of type RG, you must reorganize the physical file member referred to in the journal entry. Use the same options that were originally specified on the reorganize request when the journal entry was recorded in the journal receiver. Resume removing journal changes by starting with the journal entry that follows the 'F RG' reorganize physical file member journal entry.

»

When you apply or remove journaled changes you also have the option to have the system send information about the operation to an output file. You can specify whether information is sent about all objects in the operation or only objects that have errors. To specify that the system sends information to an output file use the Output (OUTPUT) option on the APYJRNCHG, APYJRNCHGX, or RMVJRNCHG commands.

The APYJRNCHG, APYJRNCHGX, and RMVJRNCHG commands send an escape message and ends the operation if any required journal receiver defined by the RCVRNG parameter is not on the system and associated with the journal. Use the WRKJRNA command to select the Work with journal receiver directory display, to see which journal receivers are on the system and associated with the journal. The escape message contains the name of the required journal receiver if the reason code of message CPF7053 is 1 or if message CPF9801 is sent.

«

When the processing of applying or removing journaled changes ends with an escape message, the objects can be partially changed. To determine how many changes were applied or removed for each object do one of the following:

- Review the diagnostic messages in the job log prior to the final escape message for each object.
- Use the DSPJRN command to display the journal entries indicating completion of the command.
- » If you specified to have the system send information to an output file, review the output file. The output file contains a record for each object that was processed. You can view that object's record to determine if processing completed successfully for that object.«

The command completion journal entries by object type are as follows:

Database physical file members
F journal code and an entry type of AY or RC
D journal code and entry type of DD

Integrated file system objects
B journal code and entry type of AJ

Data area objects
E journal code and entry type of EQ or EX

The **Count** field in the journal entry contains the number of journal entries that are applied or removed.

»

The system puts out a maximum of 512 diagnostic messages from Apply or Remove Journaled Changes. Therefore, it is recommended that you create an output file to determine how many changes were applied or removed for each object. For more information about the output file, see Use the apply and remove journaled changes output file.

«

Example: Apply journaled changes: The following are examples of the Apply Journaled Changes (APYJRNCHG) command applied to a database physical file, integrated file system object, and data area.

The following examples show database physical files, data areas, and integrated file system objects being processed separately. However, you can use one APYJRNCHG command if you use the OBJ parameter for files and data areas, and the OBJPATH parameter for the integrated file system objects on one command call.

» **All journaled objects**

This example restores all objects that are journaled to the journal JRN2 in one apply operation. For this example, assume that journal JRN2 is using the receiver size-option RCVSIZOPT(*MAXOPT3). Since the ending sequence number is greater than 9 999 999 999, the TOENTLRG parameter is required. The example starts applying journaled changes from the last save of the objects, to entry sequence number 500 000 000 000.

By default, the system honors the commitment boundaries. So if there is an object whose commitment boundary ends after sequence number 500 000 000 000, the apply operation ends for that object. The apply operation continues for the other objects that are journaled to the journal.

```
      APYJRNCHG JRN(JRN2) OBJ(*ALLJRNOBJ)
                FROMENT(*LASTSAVE) TOENTLRG(500000000000)
                RCVRNG(*LASTSAVE)
```

«

Database physical file

» The following command applies the changes in journal JRNA to the all the members of all files in the library DSTPRODLIB that are being journaled to journal JRNA.

```
APYJRNCHG JRN(JRNLIB/JRNA) FILE((DSTPRODLIB/*ALL))
          FROMENTLRG(*LASTSAVE) TOENTLRG(*LASTRST)
```

Because the RCVRNG parameter is not specified, the system determines the range of journal receivers to use as a result of the save information for the files. The FROMENTLRG parameter defaults to apply the changes that begin with the first journal entry after the save of the object. The required earliest receiver is the receiver that contains the D DW journal entry indicating the start of save for file DSTPRODLIB.

If the file was last saved with the save-while-active function, the saved copy of each file member includes all object-level changes in the journal entries up to the corresponding F SS journal entry. In this case, the system applies changes that begin with the first journal entry that follows the F SS entry.

If the file was last saved when it was not in use (normal save), the saved copy of each member includes all object-level changes in the journal entries up to the corresponding F MS member saved journal entry. In this case, the system applies changes that begin with the first journal entry that follows the F MS entry.
«

The following command applies the changes to the file from the journal receiver that is currently attached to the journal:

»

```
APYJRNCHG JRN(JRNLIB/JRNA) FILE((LIBA/FILEA MBR1))
          RCVRNG(*CURRENT) FROMENTLRG(*FIRST)
          TOENTLRG(*LASTRST) OUTPUT(*OUTFILE)
          OUTFILE(MYFILE) DETAIL(*ERR)
```

«

The *CURRENT journal receiver is the journal receiver that is attached to journal JRNA at the beginning of the operation. The system applies the changes from the first journal entry in this receiver to the entry before the object was last restored. Changes are applied to member MBR1 of the file FILEA.

» Because OUTPUT(*OUTFILE) is specified, an output file with the name MYFILE is created. The output file contains a record for each object, if any, for which the apply ends early because DETAIL(*ERR) is specified. «

The following command applies the changes in the journal JRNA to all members of the file FILEA beginning with the first journal entry after the file member was last saved:

```
APYJRNCHG JRN(JRNLIB/JRNA) FILE((LIBA/FILEA *ALL))
          TOJOBC(000741/USERP/WORKSTP)
```

The operation continues until the specified job closes any of the members in the file that it opened. The operation is not restricted only to those journal entries that are recorded by the specified job.

Note: This example works only if you do not specify OMTJRNE (*OPNCLO) when starting journaling for the file and you did not specify RCVSIZOPT(*MINFIXLEN) or you did not use a FIXLENDTA option that would have omitted the job name for the journal at any time while the file was journaled).

Integrated file system object

The following command applies the changes in journal JRNA to the objects in the directory MyDirectory, and its subdirectories, that are being journaled to journal JRNA:

```
APYJRNCHG JRN(JRNLIB/JRNA) OBJPATH(('/MyDirectory')) SUBTREE(*ALL)
```

» Because the RCVRNG parameter is not specified, the system determines the range of journal receivers to use as a result of the save information for the objects. Because the FROMENT or the FROMENTLRG parameters are not specified, the system applies the changes that begin with the journal entry for the last save of each of the objects.

If the object was last saved with the save-while-active function, the saved copy of each object includes all changes in the journal entries up to the corresponding B FW journal entry. In this case, the system applies changes that begin with the first journal entry that follows the B FW entry. «

If the object was last saved when it was not in use (normal save), the saved copy of each object includes all changes in the journal entries up to the corresponding B FS saved journal entry. In this case, the system applies changes that begin with the first journal entry that follows the B FS entry.

Data area

The following command applies the changes to the data area DATA1 from the journal receiver that is currently attached to the journal:

»

```
APYJRNCHG JRN(JRNLIB/JRNA) OBJ((LIBA/DATA1 *DTAARA))
          RCVRNG(*CURRENT) FROMENTLRG(*FIRST)
          TOENTLRG(*LASTRST)
```

«

The *CURRENT journal receiver is the journal receiver that is attached to journal JRNA at the beginning of the operation. The system applies the changes from the first journal entry in this receiver to the entry before the object was last restored. Changes are applied to data area DATA1.

Note: Read the Code example disclaimer for important legal information.

Example: Remove journaled changes: Even though the following examples show database physical files and data areas being processed separately, you can do them with one Remove Journaled Changes (RMVJRNCHG) command if you use the OBJ parameter for both object types.

Database physical file

The following command removes the changes in journal JRNA from the all the members of FILEA:

```
RMVJRNCHG JRN(JRNLIB/JRNA) FILE(DSTPRODLIB/FILEA)
          FROMENT(*LAST) TOENT(*FIRST)
          RCVRNG(*CURRENT)
```

The *CURRENT journal receiver is the journal receiver that is attached to journal JRNA at the beginning of the operation. The system starts removing the changes beginning with the latest entry for that member in this receiver and continues to the earliest entry for that member in this receiver.

The following command removes the changes in journal JRNA from all the members of FILEA:

»

```
RMVJRNCHG JRN(JRNLIB/JRNA) FILE(DSTPRODLIB/FILEA)
       FROMENT(*LAST) TOENT(*FIRST)
          RCVRNG(JRNLIB/RCVA10 JRNLIB/RCVA8)
          OUTPUT(*OUTFILE) OUTFILE(MYFILE)
```

«

The system starts removing the changes beginning with the last entry (the latest entry) for that member in journal receiver RCVA10 and continues to the first entry (the earliest entry) for that member on journal receiver RCVA8.

» Because OUTPUT(*OUTFILE) is specified, an output file with the name MYFILE is created. The output file contains a record for each object that the remove operation processes. See Use the apply and remove journaled changes output file for an explanation of each field in the record. «

Data area

The following removes the changes in JRNA from data area DATA1 from the last save entry to entry number 1003.

```
RMVJRNCHG JRN(JRNLIB/JRNA) OBJ((LIBA/DATA1 *DTAARA))
          RCVRNG(*CURRENT) FROMENT(*LASTSAVE) TOENT(1003)
```

If the last save operation used the save-while-active function, the system starts by removing changes from the entry preceding the last E EW start of save entry. If the last save operation was a normal save operation, the system starts by removing changes from the entry that precedes the last E ES data area saved entry. In the example, journaled changes are removed back to entry 1003.

Note: Read the Code example disclaimer for important legal information.

Example: Recover objects with partial transactions: »

If you restore an object that was saved with a save-while-active operation that specified that the object can be saved before it reaches a commitment boundary, it can have partial transactions. To recover objects that are in a partial state you must perform an apply or a remove journaled changes operation.

Another reason that an object can have partial transactions is if a long-running rollback was forced to end. However, if an object has partial transactions because of a long-running rollback, you cannot recover it with an apply or remove journaled changes operation.

If you perform save-while-active operations that can result in objects that are saved with partial transaction, it is recommended that you use Backup, Recovery, and Media Services (BRMS). You can use BRMS to automate your backup and recovery operations. BRMS automatically applies changes to objects with partial transactions and restores them to a usable state. For more detailed information see the BRMS topic.

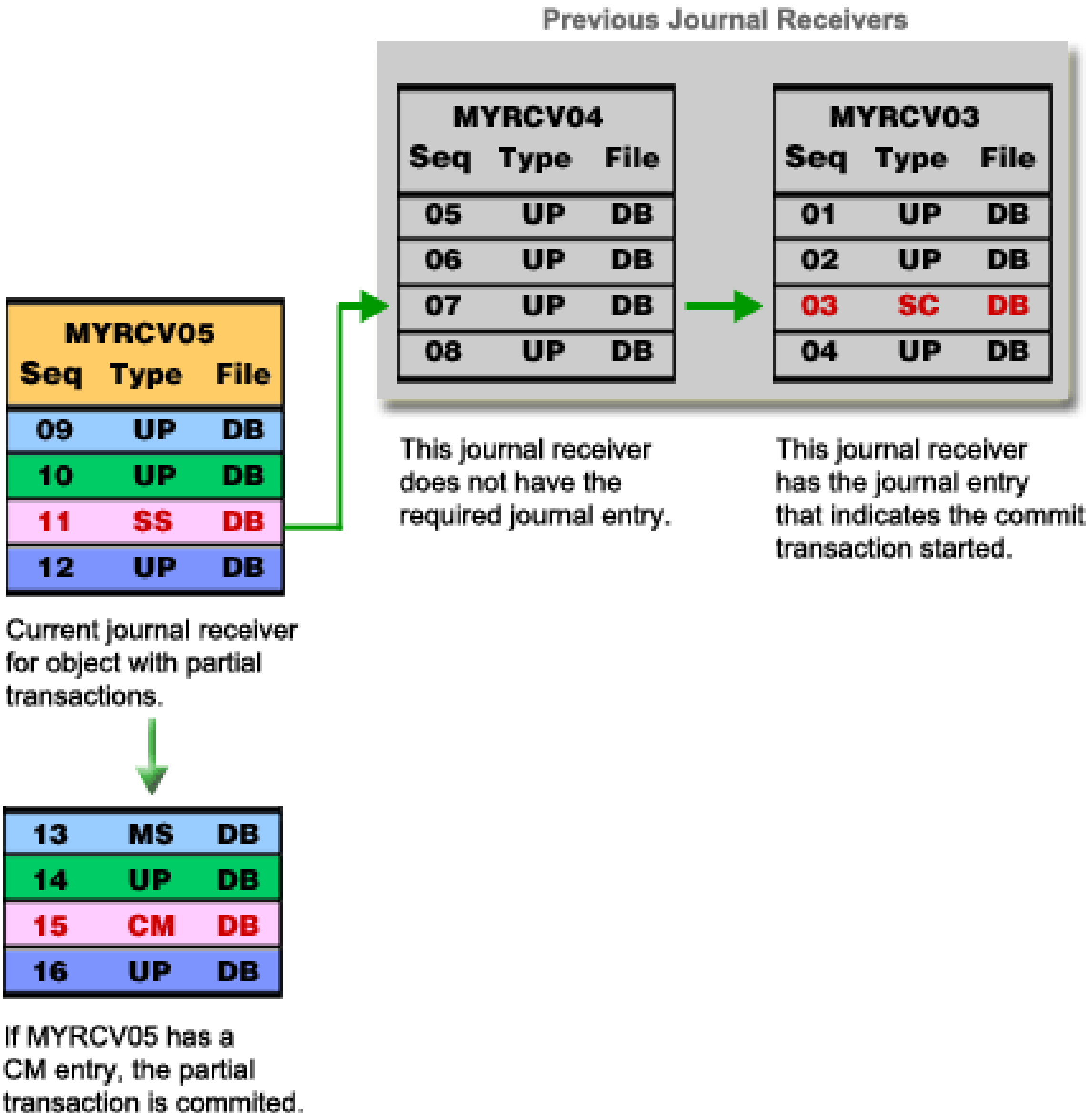
Required journal receivers

When you recover objects with partial transactions, all of the journal receivers that are required for the recovery operation must be on the system. The recovery operation might require more journal receivers than just the last one you detached. The system looks for the last journal receiver with an journal entry for the object that indicates one of the following:

- The last regular save
- The last save-while-active in which the object was saved without any partial transactions
- The earliest SC (start commit) entry for any open transactions that affect the saved object for a save with partial transactions

The following illustration demonstrates these requirements.

1. Starting with receiver MYRCV05 the apply journaled changes operation starts.
2. The systems finds the SS entry that indicates the object was saved with partial transactions.
3. If journal receiver MYRCV05 has the CM entry that indicates the transaction for the object was committed, the apply journaled changes operation applies the changes.
4. If journal receiver MYRCV05 does not have the CM entry, the system looks back to previous journal receivers.
5. Since the SC entry is not in MYRCV04, the system looks in MYRCV03.
6. The system finds the SC entry in MYRCV03 and the transaction is rolled back to that point.



MYRCV04		
Seq	Type	File
05	UP	DB
06	UP	DB
07	UP	DB
08	UP	DB

MYRCV03		
Seq	Type	File
01	UP	DB
02	UP	DB
03	SC	DB
04	UP	DB

MYRCV05		
Seq	Type	File
09	UP	DB
10	UP	DB
11	SS	DB
12	UP	DB

This journal receiver does not have the required journal entry.

This journal receiver has the journal entry that indicates the commit transaction started.

Current journal receiver for object with partial transactions.

13	MS	DB
14	UP	DB
15	CM	DB
16	UP	DB

If MYRCV05 has a CM entry, the partial transaction is commited.

As the previous figure shows, even if you are performing an apply journaled changes operation, it is still possible that the transaction can be rolled back and you will need previous journal receivers.

Restore examples

All of the objects in the following examples are database files. The following are examples of recovering objects in a partial state from three possible scenarios:

- Restore of a single object with partial transactions
- Restore of multiple objects with partial transactions resulting from a rollback that was forced to end
- Remove the partial transactions from an object that was restored with partial transactions.

Restore of a single object with partial transactions

In this example, an object, OBJ1 in library LIB1 was saved with a save-while-active operation while it had pending transactions. The save-while-active operation is the object’s most recent save. Journaled changes start from the last save and end at the last sequence number in the journal receiver.

The following is an example of the APYJRNCHG command. The default value for FROMENT is *LASTSAVE. The TOENT parameter uses the *LASTRST value to apply journaled changes up to the journal entry when the object was last restored.

```
      APYJRNCHG JRN(JRN1) FILE(LIB1/OBJ1)
                FROMENT(*LASTSAVE) TOENT(*LASTRST)
                RCVRNG(*LASTSAVE)
```

The following is an example of the RMVJRNCHG command. The following command removes the changes in journal JRN1 from the all the members of OBJ1:

```
RMVJRNCHG JRN(JRNA1) FILE(LIB1/OBJ1)
          FROMENT(*LASTSAVE) TOENT(*COMMITSTART)
          RCVRNG(*LASTSAVE)
```

Starting with the last save journal entry, only changes for journal entries for any partial transactions are removed, back to the start of the commit transaction

Remove partial transaction status from an object with partial transactions

This example uses the Change Journaled Object (CHGJRNOBJ) command because the journal receivers are not available to perform an apply or remove journaled changes operation. The Partial Transactions (PTLTNS) parameter allows the object to be used, but does not complete the transactions. The object, BRKNOBJ, still has changes caused by the partial transactions, but you are able to open the file.

Attention: Only use the following command as a last resort. You **will lose data** if you use this command. You should only use this command for the following reasons:

- You have objects with partial transaction as a result of the termination of a long-running rollback and you have no saved version to restore.
- You have objects with partial transactions as a result of a save-while-active operation, and the journal receivers required to apply or remove journaled changes have been lost, destroyed, or damaged beyond repair.

```
      CHGJRNOBJ OBJECT(LIB1/BRKNOBJ *FILE) ATR(*PTLTNS) PTLTNS(*ALWUSE)
```

Note: Read the Code example disclaimer for important legal information.

«

Journal entry information

The system creates different types of journal entries in the journal receiver for different kinds of activities. You cannot access the information in journal receivers directly. Several system commands provide formatted information from a journal receiver:

- Use the Display Journal (DSPJRN) command to display entries, print them, or write them to an output file.
- Use the Receive Journal Entry (RCVJRNE) command to specify an exit program. When entries are added to the journal receiver, they are also passed to the exit program. The exit program can, for example, write entries to save media or send them to another system.
- Use the Retrieve Journal Entry (RTVJRNE) command to retrieve journal entries to a CL program.
- Use the Retrieve Journal Entries (QjoRetrieveJournalEntries) API to retrieve journal entries into a high level language program.

When the system formats journal entries for you with the DSPJRN and RTVJRNE commands, it uses one of several layouts. These layouts include a fixed-length portion and a variable-length portion. The variable-length portion includes entry-specific data and null value indicators, if applicable. The fixed-length portion of the journal entry appears as separate fields in these layouts.

Journal entry information finder
The Journal code finder shows all the journal codes and entry types for journal entries. You can search for individual codes, display codes by category, or display all journal codes.

Journal code descriptions
This topic provides a description for all of the journal codes and categories.

Fixed-length portion of the journal entry
This topic provides the layouts of the fixed-length portion of the journal entries.

Variable-length portion of the journal entry
This topic provides the layouts of the variable-length portion of the journal entries.

Work with journal entry information
This topic provides ways that you can display, retrieve, and receive journal entries.

For information about which journal codes are affected by applying or removing journaled changes see Actions of applying or removing journaled changes by journal code.

Journal code descriptions

Following are descriptions of all the possible journal codes or categories of journal entries.

Journal Code A - System Accounting Entry
Journal entries with a journal code of A contain information about job accounting. See Work Management on the V5R1 Supplemental Manuals Web site for a detailed description of the contents of converted journal entries with journal code A.

Journal Code B- Integrated File System
Journal entries with a journal code of B contain information about changes to integrated file system objects. The only integrated file system objects that are supported are those with an object of type *STMF, *DIR or *SYMLNK. These objects must be in the ″root″(/), QOpensys, and User-defined file systems. See the Integrated file system topic for more information about file systems.

Journal Code C - Commitment Control Operation
Journal entries with a journal code of C contain commitment control information.

Journal Code D - Database File Operation
Journal entries with a journal code of D contain file level information about changes for a physical file, not an individual member.

Journal Code E - Data Area Operation
Journal entries with a journal code of E contain information about changes to journaled data areas. See Work Management on the V5R1 Supplemental Manuals Web site for more information about data areas.

Journal Code F - Database File Member Operation
Journal entries with a journal code of F contain file level information about changes for a physical file member that are being journaled to this journal. (If you use a logical file in a program, the file level information reflects the physical file on which the logical file is based.) Journal entries with journal code F can also contain file level information for access paths that are associated with physical or logical file members that are being journaled to this journal.

Journal Code I - Internal Operation
Journal entries with a journal code of I contain information about access paths or indexes or other internal operations. Entries with a journal code of I are displayed only if JRN(*INTSYSJRN) is specified or INCHIDENT(*YES) is specified on the DSPJRN command.

Journal Code J - Journal or Receiver Operation
Journal entries with a journal code of J contain information about the journal and the journal receivers.

Journal Code L - License Management
Journal entries with a journal code of L contain information about license management, such as changes to the usage limit and usage limit violations.

Journal Code M - Network Management Data
Journal entries with a journal code of M contain information about Network Management, including TCP/IP. For a description of the TCP/IP entries, see TCP/IP Configuration and Reference . For a description of the Network Management entries, see Simple Network Management Protocol (SNMP) Support .

Journal Code O - Object-Oriented Entry
Journal entries with a journal code of O contain object-oriented information. These entries are reserved for future use.

Journal Code P - Performance Tuning Entry
Journal entries with a journal code of P contain information about performance. For the description of the layout of these entries, see Work Management on the V5R1 Supplemental Manuals Web site.

Journal Code Q - Data Queue Operation
Journal entries with a journal code of Q contain information about changes to journaled data queues. See CL Programming for more information about data queues.

Journal Code R - Operation on Specific Record Journal entries with a journal code of R contain information about a change to a specific record in the physical file member that is being journaled to the journal. For a given physical file member, the record-level journal entries appear in the journal in the order that the changes were made to the file.

Journal Code S - Distributed Mail Services
Journal entries with a journal code of S contain information about SNA distribution services (SNADS), X.400(R), and mail server framework. For the description of the layout of these entries, refer to these books:

- SNA Distribution Services on the V5R1 Supplemental Manuals Web site
- AnyMail/400 Mail Server Framework Support

Journal Code T - Audit Trail Entry
Journal entries with a journal code of T contain auditing information. For the description of the layout of audit journal entries, see iSeries(TM) Security Reference .

Journal Code U - User-Generated Entry
Journal entries with a code of U are sent to the journal receiver by the Send Journal Entry (SNDJRNE) command or by the Send Journal Entry (QJOSJRNE) API. Send your own journal entries provides more information.

Journal entries by code and type

Journal entries by journal code and type

Journal Code	Entry Type	Description	Notes
A	DP	Direct print information	See Work Management on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data.
A	JB	Job resource information	See Work Management on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data.
A	SP	Spooled print information	See Work Management on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data.
B	AA	Change audit attribute	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	AJ	Start of apply	
B	AT	End of apply	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	BD	Integrated file system object deleted	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	B0	Begin create	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	B1	Create summary	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
B	B2	Link to existing object	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	B3	Rename, move object	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	B4	Remove link (parent directory)	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	B5	Remove link (link)	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	B6	Bytes cleared, after-image	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	CS	Integrated file system object closed	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	ET	End journaling for object	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
B	FA	Integrated file system object attribute changed	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	FC	Integrated file system object forced	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	FF	Storage for object freed	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	FR	Integrated file system object restored	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	FS	Integrated file system object saved	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
B	FW	Start of save for save-while-active	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	JA	Change journaled object attribute	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	JT	Start journaling for object	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	OA	Change object authority	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	OF	Integrated file system object opened	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	OG	Change primary group	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	OI	Object in use at abnormal end	See the layout for the variable width portion of this journal entry.
B	OO	Change object owner	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
B	RN	Rename file identifier	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	TR	Integrated file system object truncated	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
B	WA	Write, after-image	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QP0LJRNL.H. See the layout for the variable width portion of this journal entry.
C	BA	Commit in use at abnormal end	See the layout for the variable width portion of this journal entry.
C	BC	Start commitment control (STRCMTCTL)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
C	CM	Set of record changes committed (COMMIT)	See the layout for the variable width portion of this journal entry.
C	CN	Rollback ended early	See the layout for the variable width portion of this journal entry.
C	DB	Internal entry	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
C	EC	End commitment control (ENDCMTCTL)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
C	LW	A logical unit of work (LUW) has ended	See the layouts for the Logical Unit of Work journal entry and the following: • Header record. • Local record. • API record. • DDL record. • RMT record. • DDM record.

Journal Code	Entry Type	Description	Notes
C	PC	Prepare commit block	
C	R1	Rollback started	
C	RB	Set of record changes rolled back (ROLLBACK)	See the layout for the variable width portion of this journal entry.
C	SB	Start of savepoint	This is the start of the savepoint or nested commit cycle where it is written to the journal and occurs when the application creates an SQL SAVEPOINT. The system can also create an internal nested commit cycle to handle a series of database functions as a single operation. The entry-specific data for this journal entry is all internal data.
C	SC	Commit transaction started	
C	SQ	Release of savepoint	This is the release of the savepoint or commit of nested commit cycle. Entries are written to the journal when the application releases an SQL SAVEPOINT or when the system commits an internal nested commit cycle. See the layout for the variable width portion of this journal entry.
C	SU	Rollback of save point	This is the release of the savepoint or commit of nested commit cycle. Entries are written to the journal when the application releases an SQL SAVEPOINT or when the system commits an internal nested commit cycle. See the layout for the variable width portion of this journal entry.
D	AC	Add referential integrity constraint	See the layout for the variable width portion of this journal entry.
D	CG	Change file	See the layout for the variable width portion of this journal entry.
D	CT	Create database file	See the layout for the variable width portion of this journal entry.
D	DC	Remove referential integrity constraint	See the layout for the variable width portion of this journal entry.
D	DD	End of apply or remove	See the layout for the variable width portion of this journal entry.
D	DF	File was deleted	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
D	DG	Start of apply or remove	
D	DH	File saved	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.
D	DJ	Change journaled object attribute	See the layout for the variable width portion of this journal entry.
D	DT	Delete file	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
D	DW	Start of save-while-active save	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.
D	DZ	File restored	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
D	EF	Journaling for a physical file ended (ENDJRNPF)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
D	FM	File moved to a different library (MOVOBJ or RNMOBJ OBJTYPE(*LIB))	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
D	FN	File renamed (RNMOBJ)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
D	GC	Change constraint	See the layout for the variable width portion of this journal entry.
D	GO	Change owner	See the layout for the variable width portion of this journal entry.
D	GT	Grant authority	See the layout for the variable width portion of this journal entry.
D	ID	File in use	See the layout for the variable width portion of this journal entry.
D	JF	Journaling for a physical file started (STRJRNPF (JRNPF))	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
D	MA	Member added to file	
D	RV	Revoke authority	See the layout for the variable width portion of this journal entry.
D	TC	Add trigger	See the layout for the variable width portion of this journal entry.
D	TD	Remove trigger	See the layout for the variable width portion of this journal entry.
D	TG	Change trigger	See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
D	TQ	Refresh table	See the layout for the variable width portion of this journal entry.
E	EA	Update data area, after image	Neither the before-image nor after-image is deposited into the journal if the after-image is exactly the same as the before-image. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	EB	Update data area, before image	Neither the before-image nor after-image is deposited into the journal if the after-image is exactly the same as the before-image. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	ED	Data area deleted	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
E	EG	Start journal for data area	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
E	EH	End journal for data area	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
E	EI	Data area in use	
E	EK	Change journaled object attribute	See the layout for the variable width portion of this journal entry.
E	EL	Data area restored	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
E	EM	Data area moved	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	EN	Data area renamed	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	EQ	Data area changes applied	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	ES	Data area saved	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	EU	Remove journaled changes (RMVJRNCHG) command started	
E	EW	Start of save for data area	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	EX	Data area changes removed	The entry-specific data for these journal entries is laid out in the QSYSINC include file, QWCJRNL.H. See the layout for the variable width portion of this journal entry.
E	EY	Apply journaled changes (APYJRNCHG) command started	
F	AY	Journaled changes applied to a physical file member (APYJRNCHG)	See the layout for the variable width portion of this journal entry.
F	CB	Physical file member changed	
F	CE	Change end of data for physical file member	See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
F	CH	Change file	As of V5R1M0, the journal entry D CG is also being sent for the change file operations. IBM(TM) strongly recommends that you do your processing based on the D CG entry instead of the F CH entry because the F CH entry may be retired in a future release.
F	CL	Physical file member closed (for shared files, a close entry is made for the last close operation of the file)	See the layout for the variable width portion of this journal entry.
F	CR	Physical file member cleared (CLRPFM)	
F	C1	Rollback ended early	See the layout for the variable width portion of this journal entry.
F	DE	Physical file member deleted record count	
F	DM	Delete member	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
F	EJ	Journaling for a physical file member ended (ENDJRNPF)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
F	EP	Journaling access path for a database file member ended (ENDJRNAP)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
F	FD	Physical file member forced (written) to auxiliary storage	See the layout for the variable width portion of this journal entry.
F	FI	System-generated journal entry format information	
F	IU	Physical file member in use at the time of abnormal system end	See the layout for the variable width portion of this journal entry.
F	IZ	Physical file member initialized (INZPFM)	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. See the layout for the variable width portion of this journal entry.
F	JC	Change journaled object attribute	See the layout for the variable width portion of this journal entry.
F	JM	Journaling for a physical file member started (STRJRNPF)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
F	JP	Journaling access path for a database file member started (STRJRNAP)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
F	MC	Create member	See the layout for the variable width portion of this journal entry.
F	MD	Physical file member deleted. This entry is created when you remove the member (RMVM) or delete the file (DLTF) containing the member.	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
F	MF	Physical file member saved with storage freed (SAVOBJ, SAVCHGOBJ, or SAVLIB)	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal.
F	MM	Physical file containing the member moved to a different library (MOVOBJ or RNMOBJ OBJTYPE(*LIB))	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
F	MN	Physical file containing the member renamed (RNMM or RNMOBJ)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
F	MO	Allow use with partial transactions	See the layout for the variable width portion of this journal entry.
F	MR	Physical file member restored (RSTOBJ or RSTLIB)	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.
F	MS	Physical file member saved (SAVOBJ, SAVLIB, or SAVCHGOBJ)	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.
F	OP	Physical file member opened (for shared files, an open entry is added for the first open operation for the file)	See the layout for the variable width portion of this journal entry.
F	PD	Database file member's access path deleted (this entry is created when you remove the member (RMVM) or delete the file (DLTF) containing the member)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The object name for this entry might be misleading. It is the original name the path had when journaling started. The name is not updated if the access path is moved, renamed, or if it is implicitly shared by another logical file. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
F	PM	The logical owner of a journaled access path was moved (MOVOBJ or RNMOBJ OBJTYPE(*LIB))	After you have installed V4R2M0 or a later release, this journal type is no longer generated. See the layout for the variable width portion of this journal entry.
F	PN	The logical owner of a journaled access path was renamed (RNMOBJ or RNMM)	After you have installed V4R2M0 or a later release, this journal type is no longer generated. See the layout for the variable width portion of this journal entry.
F	RC	Journaled changes removed from a physical file member (RMVJRNCHG)	See the layout for the variable width portion of this journal entry.
F	RG	Physical file member reorganized (RGZPFM)	See the layout for the variable width portion of this journal entry.
F	RM	Member reorganized	
F	SA	The point at which the APYJRNCHG command started running	
F	SR	The point at which the RMVJRNCHG command started running	
F	SS	The start of the save of a physical file member using the save-while-active function	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.
I	DA	Directory in use at abnormal end	See the layout for the variable width portion of this journal entry.
I	DK	Internal entry	
I	IB	Internal recovery	
I	IC	Access path recovery	
I	IE	Directory recovery	
I	IF	Access path recovery	
I	IG	Access path recovery	
I	IH	Access path recovery	
I	II	Access path in use	
I	IV	Access path recovery	
I	IW	Access path recovery	
I	IX	Access path recovery	
I	IY	Access path recovery	
I	UE	Unknown entry type	
J	CI	Journal caching started	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	CX	Journal caching ended	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.

Journal Code	Entry Type	Description	Notes
J	EZ	End journaling for journal receiver	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	IA	System IPL after abnormal end	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	IN	System IPL after normal end	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	JI	Journal receiver in use at abnormal end	See the layout for the variable width portion of this journal entry.
J	JR	Start journaling for journal receiver	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	KR	Keep journal receivers for recovery	
J	LA	Activate local journal	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	LI	Inactivate local journal	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	NK	Do not keep journal receivers for recovery	
J	NR	Identifier for the next journal receiver (the receiver that was attached when the indicated receiver was detached)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
J	PR	Identifier for the previous journal receiver (the receiver that was detached when the indicated receiver was attached)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
J	RD	Deletion of a journal receiver (DLTJRNRCV)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. See the layout for the variable width portion of this journal entry.
J	RF	Storage for a journal receiver freed (SAVOBJ, SAVCHGOBJ, or SAVLIB)	See the layout for the variable width portion of this journal entry.
J	RR	Restore operation for a journal receiver (RSTOBJ or RSTLIB)	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
J	RS	Save operation for a journal receiver (SAVOBJ, SAVCHGOBJ, or SAVLIB)	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. See the layout for the variable width portion of this journal entry.
J	SI	Enter journal state (*STANDBY)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	SL	Severed link	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. This is the start of the savepoint or nested commit cycle where it is written to the journal and occurs when the application creates an SQL SAVEPOINT. The system can also create an internal nested commit cycle to handle a series of database functions as a single operation. The entry-specific data for this journal entry is all internal data.
J	SX	Exit journal state (*STANDBY)	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	UA	User independent auxiliary storage pool vary on abnormal	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	UN	User independent auxiliary storage pool vary on normal	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
J	XP	Internal entry	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver.
L	LK	License key is not valid	See the layout for the variable width portion of this journal entry.
L	LL	Usage limit changed	See the layout for the variable width portion of this journal entry.
L	LU	Usage limit exceeded	See the layout for the variable width portion of this journal entry.
M	MP	Modification of QoS policies	
M	SN	Simple Network Management Protocol (SNMP) information	See Simple Network Management Protocol (SNMP) Support for information about the entry specific data for SNMP journal entries.
M	TF	IP filter rules actions	Refer to TCP/IP Configuration and Reference for information about the entry specific data for TCP/IP journal entries.
M	TN	IP NAT rules actions	Refer to TCP/IP Configuration and Reference for information about the entry specific data for TCP/IP journal entries.

Journal Code	Entry Type	Description	Notes
M	TS	Virtual private networking (VPN) information	Refer to TCP/IP Configuration and Reference for information about the entry specific data for TCP/IP journal entries.
O	AI	Update, after image	
O	BI	Update, before image	
O	XA	Allocate object	
O	XB	Bundled entries	
O	XD	Deallocate object	
O	XI	Index operation	
O	XS	Synchronization	
O	XT	Transaction state change	
P	TP	Performance shared pool change	See Work Management on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data.
Q	QB	Start data queue journaling	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QC	Data queue cleared, no key	The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. This entry only has entry-specific data which the system uses for internal processing. There is no structure for it in the QSYSINC include file, QMHQJRNL.H
Q	QD	Data queue deleted	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. There is no entry-specific data for this entry.
Q	QE	End data queue journaling	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. There is no entry-specific data for this entry.
Q	QI	Queue in use at abnormal end	There is no entry-specific data for this entry.
Q	QJ	Data queue cleared, has key	The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
Q	QK	Send data queue entry, has key	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QL	Receive data queue entry, has key	The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QM	Data queue moved	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QN	Data queue renamed	Even if this journal has a journal state of *STANDBY, this entry type will still be deposited in the journal receiver. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QR	Receive data queue entry, no key	The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. This entry only has entry-specific data which the system uses for internal processing. There is no structure for it in the QSYSINC include file, QMHQJRNL.H
Q	QS	Send data queue entry, no key	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
Q	QX	Start of save for data queue	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QY	Data queue saved	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	QZ	Data queue restored	These entries do not indicate that they occurred as the result of a trigger program, even if a trigger program caused the event. That information is not available at the time the entry is written to the journal. The entry-specific data for data queue journal entries are laid out in the QSYSINC include file, QMHQJRNL.H. See the layout for the variable width portion of this journal entry.
Q	VE	Internal entry	This is an internal entry. No layout of entry-specific data is provided.
Q	VQ	Internal entry	This is an internal entry. No layout of entry-specific data is provided.
R	BR	Before-image of record updated for rollback operation	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
R	DL	Record deleted in the physical file member	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. See the layout for the variable width portion of this journal entry.
R	DR	Record deleted for rollback operation	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. See the layout for the variable width portion of this journal entry.
R	IL	Increment record limit	These entries have entry-specific data which the system uses for internal processing.
R	PT	Record added to a physical file member. If the file is set up to reuse deleted records, then you may receive either a PT or PX journal entry for the change	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. See the layout for the variable width portion of this journal entry.
R	PX	Record added directly by RRN (relative record number) to a physical file member. If the file is set up to reuse deleted records, then you may receive either a PT or PX journal entry for the change	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
R	UB	Before-image of a record that is updated in the physical file member (this entry is present only if IMAGES(*BOTH) is specified on the STRJRNPF command)	Neither the before-image nor after-image is deposited into the journal if the after-image is exactly the same as the before-image. This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. See the layout for the variable width portion of this journal entry.
R	UP	After-image of a record that is updated in the physical file member	Neither the before-image nor after-image is deposited into the journal if the after-image is exactly the same as the before-image. This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. See the layout for the variable width portion of this journal entry.
R	UR	After-image of a record that is updated for rollback information	This journal entry may have data which can only be accessed by using either the QjoRetrieveJournalEntries API or the RCVJRNE command. For the RCVJRNE command, use the ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) parameters. In all other interfaces, if the data is not visible, the incomplete data indicator will be on and *POINTER will appear in the Entry Specific Data. For more information, refer to Work with pointers in journal entries. This entry may have minimized entry specific data (ESD). It will have minimized ESD if its corresponding object type deposits minimized journal entries through the MINENTDTA parameter for this journal or journal receiver. See the layout for the variable width portion of this journal entry.

Journal Code	Entry Type	Description	Notes
S	AL	SNA alert focal point information	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS.
S	CF	Mail configuration information	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS. See AnyMail/400 Mail Server Framework Support for the layout of the entry specific data.
S	DX	X.400$^{(R)}$ process debug entry	
S	ER	Mail error information	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS. See AnyMail/400 Mail Server Framework Support for the layout of the entry specific data.
S	LG	Mail logging table information	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS. See AnyMail/400 Mail Server Framework Support for the layout of the entry specific data.
S	MX	A change was made to X.400$^{(R)}$ MTA configuration	
S	NX	A change was made to X.400$^{(R)}$ delivery notification	
S	RT	Mail routing information	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS. See AnyMail/400 Mail Server Framework Support for the layout of the entry specific data.
S	RX	A change was made to X.400$^{(R)}$ route configuration	
S	SY	Mail system information	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS. See AnyMail/400 Mail Server Framework Support for the layout of the entry specific data.
S	UX	A change was made to X.400$^{(R)}$ user or probe	
S	XE	DSNX error entry	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS.

Journal Code	Entry Type	Description	Notes
S	XL	DSNX logging entry	See SNA Distribution Services on the V5R1 Supplemental Manuals Web site for the layout of the entry specific data for entries generated by SNADS.
S	XX	An error was detected by the X.400(R) process	
T	AD	A change was made to the auditing attribute	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	AF	All authority failures	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	AP	A change was made to program adopt	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	AU	Attribute change	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CA	Changes to object authority (authorization list or object)	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CD	A change was made to a command string	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CO	Create object	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CP	Create, change, restore user profiles	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CQ	A change was made to a change request descriptor	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CU	Cluster operation	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CV	Connection verification	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	CY	Cryptographic configuration	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	DI	Directory services	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	DO	All delete operations on the system	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	DS	DST security officer password reset	See iSeries(TM)Security Reference for the layout of the entry specific data.

Journal Code	Entry Type	Description	Notes
T	EV	Environment variable	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	GR	General purpose audit record	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	GS	A descriptor was given	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	IP	Inter-process communication event	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	IR	IP rules actions	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	IS	Internet security management	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	JD	Changes to the USER parameter of a job description	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	JS	A change was made to job data	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	KF	Key ring file name	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	LD	A link, unlink, or lookup operation to a directory	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	ML	A change was made to office services mail	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	NA	Changes to network attributes	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	ND	Directory search violations	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	NE	End point violations	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	OM	Object management change	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	OR	Object restored	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	OW	Changes to object ownership	See iSeries(TM)Security Reference for the layout of the entry specific data.

Journal Code	Entry Type	Description	Notes
T	O1	Single optical object access	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	O2	Dual optical object access	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	O3	Optical volume access	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	PA	Changes to programs (CHGPGM) that will now adopt the owner’s authority	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	PG	Changes to an object’s primary group	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	PO	A change was made to printed output	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	PS	Profile swap	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	PW	Passwords used that are not valid	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RA	Restore of objects when authority changes	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RJ	Restore of job descriptions that contain user profile names	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RO	Restore of objects when ownership information changes	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RP	Restore of programs that adopt their owner’s authority	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RQ	A change request descriptor was restored	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RU	Restore of authority for user profiles	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	RZ	The primary group for an object was changed during a restore operation	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SD	A change was made to the system directory	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SE	Changes to subsystem routing	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.

Journal Code	Entry Type	Description	Notes
T	SF	A change was made to a spooled output file	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SG	Asynchronous signals	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SK	Secure sockets connection	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SM	A change was made by system management	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SO	A change was made by server security	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	ST	A change was made by system tools	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	SV	Changes to system values	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VA	Changes to access control list	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VC	Connection started or ended	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VF	Server files were closed	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VL	An account limit was exceeded	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VN	A logon or logoff operation on the network	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VO	Actions on validation lists	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VP	A network password error	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VR	A network resources was accessed	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VS	A server session started or ended	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.
T	VU	A network profile was changed	See iSeries$^{(TM)}$Security Reference for the layout of the entry specific data.

Journal Code	Entry Type	Description	Notes
T	VV	Service status was changed	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X0	Network authentication	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X1	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X2	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X3	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X4	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X5	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X6	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X7	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X8	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	X9	Reserved for future audit entry	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	YC	A change was made to DLO change access	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	YR	A change was made to DLO read access	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	ZC	A change was made to object change access	Only one entry per opened file. The member name will not be displayed in the entry specific data for based on physical files. See iSeries(TM)Security Reference for the layout of the entry specific data.
T	ZM	An object was accessed using a method	See iSeries(TM)Security Reference for the layout of the entry specific data.
T	ZR	A change was made to Object read access	See iSeries(TM)Security Reference for the layout of the entry specific data.

Journal Code	Entry Type	Description	Notes
U		User-specified. The Entry-specific data is the value specified on the ENTDTA parameter of the SNDJRNE command or with the entry data parameter for the QJOSJRNE API	The entry is deposited in the journal receiver even if the journal state is *STANDBY, if the user chooses to override the *STANDBY state on the SNDJRNE command or QJOSJRNE API.

Fixed-length portion of the journal entry

When you use the Display Journal (DSPJRN) command, Receive Journal Entry (RCVJRNE) command, Retrieve Journal Entry (RTVJRNE) command, or the Retrieve Journal Entries (QjoRetrieveJournalEntries) API you can select one of the formats in which to receive the layout for the fixed-length portion of the journal entry:

- *TYPE1
- *TYPE2
- *TYPE3
- *TYPE4
- *TYPE5

***TYPE1 format**

The *TYPE1 format shows the fields that are common for all journal entry types. These fields are shown when you request *TYPE1 for the output file format or the entry type format.

***TYPE2 format**

If you request OUTFILFMT(*TYPE2) on the DSPJRN command, or ENTFMT(*TYPE2) on the RCVJRNE or RTVJRNE command, then the fixed-length portion of each converted journal entry is the same as the format in *TYPE1, except for the information that follows the commit cycle identifier field. The fields of the prefix that follow the commit cycle identifier are shown in *TYPE2 field descriptions.

TYPE3 field descriptions

A third value, *TYPE3, is supported on the OUTFILFMT parameter for the DSPJRN command, and the ENTFMT parameter for the RCVJRNE and RTVJRNE commands. If either OUTFILFMT(*TYPE3) is specified on the DSPJRN command or ENTFMT(*TYPE3) is specified on the RCVJRNE or RTVJRNE command, the information in the prefix portion of a converted journal entry is shown in *TYPE3 field descriptions. *TYPE3 has the same information as the *TYPE1 and *TYPE2 formats, except that it has a different date format and a null-values indicator.

***TYPE4 field descriptions**

A fourth value, *TYPE4, is supported on the OUTFILFMT parameter for the DSPJRN command and the ENTFMT parameter for the RCVJRNE and RTVJRNE commands. If either OUTFILFMT(*TYPE4) is specified on the DSPJRN command or ENTFMT(*TYPE4) is specified on the RCVJRNE or RTVJRNE command, the information in the prefix portion of a converted journal entry is shown in Table 4. *TYPE4 output includes all of the *TYPE3 information, plus information about journal identifiers, triggers, and referential constraints and entries which will be ignored by the APYJRNCHG or RMVJRNCHG commands.

***TYPE5 field descriptions**

The *TYPE5 format is only available with the DSPJRN and RTVJRNE commands. The *TYPE5 format is supported on the OUTFILFMT parameter for the DSPJRN command and ENTFMT parameter of the RTVJRNE command. If you specify OUTFILFMT(*TYPE5) on the DSPJRN command or ENTFMT(*TYPE5) on the RTVJRNE command, the information in the prefix portion of a converted journal entry is shown in Table 5. *TYPE5 output includes all of the *TYPE4 information, plus information about the following:

- System sequence number

- Thread identifier
- Remote address
- Address family
- Remote port
- Arm number
- Receiver name
- Receiver library name
- Receiver library ASP device name
- Program library name
- Program library ASP device name
- Program library ASP number
- Logical unit of work
- Transaction identifier
- Receiver library ASP number

The RCVJRNE command also supports the *TYPEPTR and *JRNENTFMT formats. The layout of the journal entry data for the *TYPEPTR interface is the same as the RJNE0100 format which is described in the QjoRetrieveJournalEntries API.

The layout of the journal entry data for the *JRNENTFMT interface is the same as either the RJNE0100 format or the RJNE0200 format of the QjoRetrieveJournalEntries API. You can select which format to use by selecting the RJNE0100 or the RJNE0200 value for the Journal Entry Format (JRNENTFMT) parameter of the RCVJRNE command.

You can find the field descriptions for layouts *TYPE1, *TYPE2, *TYPE3, *TYPE4, and *TYPE5 in the Journal entry information finder.

Layouts for the fixed-length portion of journal entries

***TYPE1 field descriptions of the fixed-length portion of a journal entry**

These fields are shown when you request *TYPE1 for the output file format or the entry type format. The uppercase field names shown in parentheses are used in the system-supplied output file QSYS/QADSPJRN. The field names that are in italics are the variable names for these fields in the QjoRetrieveJournalEntries API header file. These variables are under the type definition for the RJNE0100 format. The QjoRetrieveJournalEntries API header is in the QJOURN.H file of the QSYSINC library.

Relative offset	Field	Format	Description
1	Entry length (JOENTL)	Zoned (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry. If the journal entry has the incomplete data indicator on, then this length does not include that additional data which could be pointed to. This length includes the length of the data that is actually returned, which includes entry specific data of up to 32 766 bytes.

Relative offset	Field	Format	Description
» 6	Sequence number (JOSEQN, *Seq_Number*)	Zoned (10,0)	Assigned by the system to each journal entry. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the sequence number is larger than 9 999 999 999. «
16	Journal code (JOCODE, *Jrn_Code*)	Char (1)	Identifies the primary category of the journal entry: **A=** System accounting entry **B=** Integrated file system operation **C=** Commitment control operation **D=** Database file operation **E=** Data area operation **F=** Database file member operation **I=** Internal operation **J=** Journal or receiver operation **L=** License management **M=** Network management data **O=** Object oriented entry **P=** Performance tuning entry **Q=** Data queue operation **R=** Operation on a specific record **S=** Distributed mail services **T=** Audit trail entry **U=** User-generated entry (added by the SNDJRNE command or QJOSJRNE API) The journal codes are described in more detail in Journal code descriptions.
17	Entry type (JOENTT, *Entry_Type*)	Char (2)	Further identifies the type of user-created or system-created entry. See the Journal code finder for descriptions of the entry types.
19	Date stamp (JODATE)	Char (6)	Specifies the system date when the entry was added and is in the format of the job attribute DATFMT. The system cannot assure that the date stamp is always in ascending order for sequential journal entries because you can change the value of the system date.

Relative offset	Field	Format	Description
25	Time stamp (JOTIME, *Time_Stamp*)	Zoned (6,0)	Corresponds to the system time (in the format hhmmss) when the entry was added. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.
31	Job name (JOJOB, *Job_Name*)	Char (10)	Specifies the name of the job that added the entry. **Notes:** 1. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the job name. 2. If the job name was not available when the journal entry was deposited, then *NONE is written for the job name.
41	User name (JOUSER, *User_Name*)	Char (10)	Specifies the user profile name of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then blanks are written for the user name.
51	Job number (JONBR, *Job_Number*)	Zoned (6,0)	Specifies the job number of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then zeroes are written for the job number.
57	Program name (JOPGM, *Program_Name*)	Char (10)	Specifies 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. **Notes:** 1. If the program that deposited the journal entry is an original program model program, this data will be complete. Otherwise, this data will be unpredictable. 2. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, *OMITTED is given for the program name.

Relative offset	Field	Format	Description
67	Object name (JOOBJ, *Object*)	Char (10)	Specifies the name of the object for which the journal entry was added.[1] This is blank for some entries. If the journaled object is an integrated file system object, then this field is the first 10 bytes of the file identifier.
77	Library name (JOLIB)	Char (10)	Specifies the name of the library containing the object[1]. If the journaled object is an integrated file system object, then the first 6 characters of this field are the last 6 bytes of the file identifier.
87	Member name (JOMBR)	Char (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
97	Count/relative record number (JOCTRR, *Count_Rel_Rec_Num*)	Zoned (10,0)	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. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • Change end of data journal entry • CHGJRN journal entries • COMMIT journal entry • INZPFM journal entry This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the count or relative record number is larger than 9 999 999 999.
107	Indicator flag (JOFLAG, *Indicator_Flag*)	Char (1)	Contains an indicator for the operation. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • COMMIT journal entry • INZPFM journal entry • IPL and in-use journal entries • Journal code R (all journal entry types except IL) • ROLLBACK journal entry • Start-journal journal entries
» 108	Commit cycle identifier (JOCCID, *Commit_Cycle_Id*)	Zoned (10,0)	Contains a number that identifies the commit cycle. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the commit cycle identifier is larger than 9 999 999 999.«

Relative offset	Field	Format	Description
118	Incomplete Data (JOINCDAT)	Char (1)	Indicates whether this entry has data that is not being retrieved for one of the following reasons: • The length of the entry-specific data exceeds 32 766 bytes. • The entry is associated with a database file that has one or more fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object). The possible values are: **0 =** This entry has all possible data **1 =** This entry has incomplete data. Any data which is marked as incomplete, can only be viewed by using either the QjoRetrieveJournalEntries API, or the command RCVJRNE with any of the following parameters: • ENTFMT(*TYPEPTR) • ENTFMT(*JRNENTFMT) • RTNPTR (with any value specified other than *NONE)
119	Minimized entry specific data (JOMINESD)	Char (1)	Indicates whether this entry has minimized entry specific data. The possible values are: **0 =** This entry has complete entry specific data. **1 =** This entry has minimized entry specific data.
120	Reserved field (JORES)	Char (6)	Always contains zeros. Contains hexadecimal zeros in the output file.

Note:

[1]If the journal receiver was attached prior to installing V4R2M0 on your system, then the following items are true:

- If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, 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 parameter, the current fully qualified name of the file appears in the converted 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.

*TYPE2 field descriptions of the fixed-length portion of a journal entry

These fields are shown when you request *TYPE2 for the output file format or the entry type format. The uppercase field names shown in parentheses are used in the system-supplied output file QSYS/QADSPJRN. The field names that are in italics are the variable names for these fields in the QjoRetrieveJournalEntries API header file. These variables are under the type definition for the RJNE0100 format. The QjoRetrieveJournalEntries API header is in the QJOURN.H file of the QSYSINC library.

Offset	Field	Format	Description
1	Entry length (JOENTL)	Zoned (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry. If the journal entry has the incomplete data indicator on, then this length does not include that additional data which could be pointed to. This length includes the length of the data that is actually returned, which includes entry specific data of up to 32 766 bytes.
» 6	Sequence number (JOSEQN, *Seq_Number*)	Zoned (10,0)	Assigned by the system to each journal entry. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the sequence number is larger than 9 999 999 999. «
16	Journal code (JOCODE, *Jrn_Code*)	Char (1)	Identifies the primary category of the journal entry: **A=** System accounting entry **B=** Integrated file system operation **C=** Commitment control operation **D=** Database file operation **E=** Data area operation **F=** Database file member operation **I=** Internal operation **J=** Journal or receiver operation **L=** License management **M=** Network management data **O=** Object oriented entry **P=** Performance tuning entry **Q=** Data queue operation **R=** Operation on a specific record **S=** Distributed mail services **T=** Audit trail entry **U=** User-generated entry (added by the SNDJRNE command or QJOSJRNE API) The journal codes are described in more detail in Journal code descriptions.

Offset	Field	Format	Description
17	Entry type (JOENTT, *Entry_Type*)	Char (2)	Further identifies the type of user-created or system-created entry. See the Journal code finder for descriptions of the entry types.
19	Date stamp (JODATE)	Char (6)	Specifies the system date when the entry was added and is in the format of the job attribute DATFMT. The system cannot assure that the date stamp is always in ascending order for sequential journal entries because you can change the value of the system date.
25	Time stamp (JOTIME, *Time_Stamp*)	Zoned (6,0)	Corresponds to the system time (in the format hhmmss) when the entry was added. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.
31	Job name (JOJOB, *Job_Name*)	Char (10)	Specifies the name of the job that added the entry. **Notes:** 1. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the job name. 2. If the job name was not available when the journal entry was deposited, then *NONE is written for the job name.
41	User name (JOUSER, *User_Name*)	Char (10)	Specifies the user profile name of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then blanks are written for the user name.
51	Job number (JONBR, *Job_Number*)	Zoned (6,0)	Specifies the job number of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then zeroes are written for the job number.

Offset	Field	Format	Description
57	Program name (JOPGM, *Program_Name*)	Char (10)	Specifies 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. **Notes:** 1. If the program that deposited the journal entry is an original program model program, this data will be complete. Otherwise, this data will be unpredictable. 2. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, *OMITTED is given for the program name.
67	Object name (JOOBJ, *Object*)	Char (10)	Specifies the name of the object for which the journal entry was added.[1] This is blank for some entries. If the journaled object is an integrated file system object, then this field is the first 10 bytes of the file identifier.
77	Library name (JOLIB)	Char (10)	Specifies the name of the library containing the object[1]. If the journaled object is an integrated file system object, then the first 6 characters of this field are the last 6 bytes of the file identifier.
87	Member name (JOMBR)	Char (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
97	Count/relative record number (JOCTRR, *Count_Rel_Rec_Num*)	Zoned (10,0)	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. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • Change end of data journal entry • CHGJRN journal entries • COMMIT journal entry • INZPFM journal entry This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the count or relative record number is larger than 9 999 999 999.

Offset	Field	Format	Description
107	Indicator flag (JOFLAG, *Indicator_Flag*)	Char (1)	Contains an indicator for the operation. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • COMMIT journal entry • INZPFM journal entry • IPL and in-use journal entries • Journal code R (all journal entry types except IL) • ROLLBACK journal entry • Start-journal journal entries
» 108	Commit cycle identifier (JOCCID, *Commit_Cycle_Id*)	Zoned (10,0)	Contains a number that identifies the commit cycle. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the commit cycle identifier is larger than 9 999 999 999. «
118	User profile (JOUSPF)	Char (10)	Specifies the name of the user profile under which the job was running when the entry was created. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the user profile.
128	System name (JOSYNM)	Char (8)	Specifies the name of the system on which the entry is being displayed, printed, retrieved, or received 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.

Offset	Field	Format	Description
136	Incomplete data (JOINCDAT)	Char (1)	Indicates whether this entry has data that is not being retrieved for one of the following reasons: • The length of the entry-specific data exceeds 32 766 bytes. • The entry is associated with a database file that has one or more fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object). The possible values are: **0 =** This entry has all possible data **1 =** This entry has incomplete data. Any data which is marked as incomplete, can only be viewed by using either the QjoRetrieveJournalEntries API, or the command RCVJRNE with any of the following parameters: • ENTFMT(*TYPEPTR) • ENTFMT(*JRNENTFMT) • RTNPTR (with any value specified other than *NONE)
137	Minimized entry specific data (JOMINESD)	Char (1)	Indicates whether this entry has minimized entry specific data. The possible values are: **0 =** This entry has complete entry specific data. **1 =** This entry has minimized entry specific data.
138	Reserved field (JORES)	Char (18)	Always contains zeros. Contains hexadecimal zeros in the output file.

Note:

[1]If the journal receiver was attached prior to installing V4R2M0 on your system, then the following items are true:

- If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, 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 parameter, the current fully qualified name of the file appears in the converted 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.

*TYPE3 field descriptions of the fixed-length portion of a journal entry

These fields are shown when you request *TYPE3 for the output file format or the entry type format. The uppercase field names shown in parentheses are used in the system-supplied output file QSYS/QADSPJRN. The field names that are in italics are the variable names for these fields in the QjoRetrieveJournalEntries API header file. These variables are under the type definition for the RJNE0100 format. The QjoRetrieveJournalEntries API header is in the QJOURN.H file of the QSYSINC library.

Offset	Field	Format	Description
1	Entry length (JOENTL)	Zoned (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry. If the journal entry has the incomplete data indicator on, then this length does not include that additional data which could be pointed to. This length includes the length of the data that is actually returned, which includes entry specific data of up to 32 766 bytes.
» 6	Sequence number (JOSEQN, *Seq_Number*)	Zoned decimal (10,0)	Assigned by the system to each journal entry. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the sequence number is larger than 9 999 999 999. «
16	Journal code (JOCODE, *Jrn_Code*)	Char (1)	Identifies the primary category of the journal entry: **A=** System accounting entry **B=** Integrated file system operation **C=** Commitment control operation **D=** Database file operation **E=** Data area operation **F=** Database file member operation **I=** Internal operation **J=** Journal or receiver operation **L=** License management **M=** Network management data **O=** Object oriented entry **P=** Performance tuning entry **Q=** Data queue operation **R=** Operation on a specific record **S=** Distributed mail services **T=** Audit trail entry **U=** User-generated entry (added by the SNDJRNE command or QJOSJRNE API) The journal codes are described in more detail in Journal code descriptions.

Offset	Field	Format	Description
17	Entry type (JOENTT, *Entry_Type*)	Char (2)	Further identifies the type of user-created or system-created entry. See the Journal code finder for descriptions of the entry types.
19	Time stamp (JOTMST)	Char (26)	Corresponds to the system date and time when the journal entry was added in the journal receiver. The time stamp is in SAA(R) format. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.
45	Job name (JOJOB, *Job_Name*)[1]	Char (10)	Specifies the name of the job that added the entry. **Notes:** 1. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the job name. 2. If the job name was not available when the journal entry was deposited, then *NONE is written for the job name.
55	User name (JOUSER, *User_Name*)	Char (10)	Specifies the user profile name of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then blanks are written for the user name.
65	Job number (JONBR, *Job_Number*)	Zoned (6,0)	Specifies the job number of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then zeroes are written for the job number.
71	Program name (JOPGM, *Program_Name*)	Char (10)	Specifies 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. **Notes:** 1. If the program that deposited the journal entry is an original program model program, this data will be complete. Otherwise, this data will be unpredictable. 2. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, *OMITTED is given for the program name.

Offset	Field	Format	Description
81	Object name (JOOBJ, *Object*)	Char (10)	Specifies the name of the object for which the journal entry was added.[1] This is blank for some entries. If the journaled object is an integrated file system object, then this field is the first 10 bytes of the file identifier.
91	Library name (JOLIB)	Char (10)	Specifies the name of the library containing the object[1]. If the journaled object is an integrated file system object, then the first 6 characters of this field are the last 6 bytes of the file identifier.
101	Member name (JOMBR)	Char (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
111	Count/relative record number (JOCTRR, *Count_Rel_Rec_Num*)	Zoned (10,0)	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. • APYJRNCHG and RMVJRNCHG journal entries • Change end of data journal entry • CHGJRN journal entries • COMMIT journal entry • INZPFM journal entry This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the count or relative record number is larger than 9 999 999 999.
121	Indicator flag (JOFLAG, *Indicator_Flag*)	Char (1)	Contains an indicator for the operation. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • COMMIT journal entry • INZPFM journal entry • IPL and in-use journal entries • Journal code R (all journal entry types except IL) • ROLLBACK journal entry • Start-journal journal entries
» 122	Commit cycle identifier (JOCCID, *Commit_Cycle_Id*)	Zoned (10,0)	Contains a number that identifies the commit cycle. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the commit cycle identifier is larger than 9 999 999 999. «

Offset	Field	Format	Description
132	User profile (JOUSPF)	Char (10)	Specifies the name of the user profile under which the job was running when the entry was created. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the user profile.
142	System name (JOSYNM)	Char (8)	Specifies the name of the system on which the entry is being displayed, printed, retrieved, or received 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.
Note: [1]If the journal receiver was attached prior to installing V4R2M0 on your system, then the following items are true: • If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, 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 parameter, the current fully qualified name of the file appears in the converted 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.			

*TYPE4 field descriptions of the fixed-length portion of a journal entry

These fields are shown when you request *TYPE4 for the output file format or the entry type format. The uppercase field names shown in parentheses are used in the system-supplied output file QSYS/QADSPJRN. The field names which are in italics are the variable names for these fields in the QjoRetrieveJournalEntries API header file. These variables are under the type definition for the RJNE0100 format. The QjoRetrieveJournalEntries API header is in the QJOURN.H file of the QSYSINC library.

Offset	Field	Format	Description
1	Entry length (JOENTL)	Zoned (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry. If the journal entry has the incomplete data indicator on, then this length does not include that additional data which could be pointed to. This length includes the length of the data that is actually returned, which includes entry specific data of up to 32 766 bytes.

Offset	Field	Format	Description
≫ 6	Sequence number (JOSEQN, *Seq_Number*)	Zoned decimal (10,0)	Assigned by the system to each journal entry. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the sequence number is larger than 9 999 999 999. ≪
16	Journal code (JOCODE, *Jrn_Code*)	Char (1)	Identifies the primary category of the journal entry: **A=** System accounting entry **B=** Integrated file system operation **C=** Commitment control operation **D=** Database file operation **E=** Data area operation **F=** Database file member operation **I=** Internal operation **J=** Journal or receiver operation **L=** License management **M=** Network management data **O=** Object oriented entry **P=** Performance tuning entry **Q=** Data queue operation **R=** Operation on a specific record **S=** Distributed mail services **T=** Audit trail entry **U=** User-generated entry (added by the SNDJRNE command or QJOSJRNE API) The journal codes are described in more detail in Journal code descriptions.
17	Entry type (JOENTT, *Entry_Type*)	Char (2)	Further identifies the type of user-created or system-created entry. See the Journal code finder for descriptions of the entry types.
19	Time stamp (JOTMST)	Char (26)	Corresponds to the system date and time when the journal entry was added in the journal receiver. The time stamp is in SAA format. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.

Offset	Field	Format	Description
45	Job name (JOJOB, *Job_Name*)[1]	Char (10)	Specifies the name of the job that added the entry. **Notes:** 1. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the job name. 2. If the job name was not available when the journal entry was deposited, then *NONE is written for the job name.
55	User name (JOUSER, *User_Name*)	Char (10)	Specifies the user profile name of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then blanks are written for the user name.
65	Job number (JONBR, *Job_Number*)	Zoned (6,0)	Specifies the job number of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then zeroes are written for the job number.
71	Program name (JOPGM, *Program_Name*)	Char (10)	Specifies 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. **Notes:** 1. If the program that deposited the journal entry is an original program model program, this data will be complete. Otherwise, this data will be unpredictable. 2. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, *OMITTED is given for the program name.
81	Object name (JOOBJ, *Object*)	Char (10)	Specifies the name of the object for which the journal entry was added.[1] This is blank for some entries. If the journaled object is an integrated file system object, then this field is the first 10 bytes of the file identifier.
91	Library name (JOLIB)	Char (10)	Specifies the name of the library containing the object[1]. If the journaled object is an integrated file system object, then the first 6 characters of this field are the last 6 bytes of the file identifier.

Offset	Field	Format	Description
101	Member name (JOMBR)	Char (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
111	Count/relative record number (JOCTRR, *Count_Rel_Rec_Num*)	Zoned (10,0)	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. • APYJRNCHG and RMVJRNCHG journal entries • Change end of data journal entry • CHGJRN journal entries • COMMIT journal entry • INZPFM journal entry This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the count or relative record number is larger than 9 999 999 999.
121	Indicator flag (JOFLAG, *Indicator_Flag*)	Char (1)	Contains an indicator for the operation. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • COMMIT journal entry • INZPFM journal entry • IPL and in-use journal entries • Journal code R (all journal entry types except IL) • ROLLBACK journal entry • Start-journal journal entries
» 122	Commit cycle identifier (JOCCID, *Commit_Cycle_Id*)	Zoned (10,0)	Contains a number that identifies the commit cycle. 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. This field can contain a -1 if receiver-size option RCVSIZOPT(*MAXOPT3) is selected and the actual value of the commit cycle identifier is larger than 9 999 999 999. «
132	User profile (JOUSPF)	Char (10)	Specifies the name of the user profile under which the job was running when the entry was created. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the user profile.
142	System name (JOSYNM)	Char (8)	Specifies the name of the system on which the entry is being displayed, printed, retrieved, or received 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.

Offset	Field	Format	Description
150	Journal identifier (JOJID)	Char(10)	Specifies 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. However, if journaling is stopped, there is no guarantee that the JID will be the same if journaling is started again for the same object. If no JID is associated with the entry, this field has hexadecimal zeros.
160	Referential constraint (JORCST)	Char(1)	Indicates whether this entry was recorded for actions that occurred on records that are part of a referential constraint. The possible values are: **0 =** This entry was not created as part of a referential constraint. **1 =** This entry was created as part of a referential constraint.
161	Trigger (JOTGR)	Char(1)	Indicates whether this entry was created as result of a trigger program. The possible values are: **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.
162	Incomplete data (JOINCDAT)	Char (1)	Indicates whether this entry has data that is not being retrieved for one of the following reasons: • The length of the entry-specific data exceeds 32 766 bytes. • The entry is associated with a database file that has one or more fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object). The possible values are: **0 =** This entry has all possible data **1 =** This entry has incomplete data. Any data which is marked as incomplete, can only be viewed by using either the QjoRetrieveJournalEntries API, or the command RCVJRNE with any of the following parameters: • ENTFMT(*TYPEPTR) • ENTFMT(*JRNENTFMT) • RTNPTR (with any value specified other than *NONE)

Offset	Field	Format	Description
163	Ignored by APYJRNCHG or RMVJRNCHG (JOIGNAPY)	Char (1)	Indicates whether this journal entry will be ignored by the execution of the APYJRNCHG or RMVJRNCHG commands, even though normally this journal entry type has an effect during those command invocations. The possible values are: **0 =** This entry is not ignored by the APYJRNCHG or RMVJRNCHG commands. **1 =** This entry is ignored by the APYJRNCHG or RMVJRNCHG commands.
164	Minimized entry specific data (JOMINESD)	Char (1)	Indicates whether this entry has minimized entry specific data. The possible values are: **0 =** This entry has complete entry specific data. **1 =** This entry has minimized entry specific data.
165	Reserved area (JORES)	Char (5)	Always contains zeros. Contains hexadecimal zeros in the output file.

Note:

[1]If the journal receiver was attached prior to installing V4R2M0 on your system, then the following items are true:

- If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, 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 parameter, the current fully qualified name of the file appears in the converted 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.

*TYPE5 field descriptions of the fixed-length portion of a journal entry

These fields are shown when you request *TYPE5 for the output file format or the entry type format. The uppercase field names shown in parentheses are used in the system-supplied output file QSYS/QADSPJRN. The field names that are italics are the variable names for these fields in the QjoRetrieveJournalEntries API header file. These variables are under the type definition for the RJNE0100 format. The QjoRetrieveJournalEntries API header is in the QJOURN.H file of the QSYSINC library.

Offset	Field	Format	Description
1	Entry length (JOENTL)	Zoned (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry. If the journal entry has the incomplete data indicator on, then this length does not include that additional data which could be pointed to. This length includes the length of the data that is actually returned, which includes entry specific data of up to 32 766 bytes.

Offset	Field	Format	Description
6	Sequence number (JOSEQN, *Seq_Number*)	Char (20)	Assigned by the system to each journal entry. 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.
26	Journal code (JOCODE, *Jrn_Code*)	Char (1)	Identifies the primary category of the journal entry: **A=** System accounting entry **B=** Integrated file system operation **C=** Commitment control operation **D=** Database file operation **E=** Data area operation **F=** Database file member operation **I=** Internal operation **J=** Journal or receiver operation **L=** License management **M=** Network management data **O=** Object oriented entry **P=** Performance tuning entry **Q=** Data queue operation **R=** Operation on a specific record **S=** Distributed mail services **T=** Audit trail entry **U=** User-generated entry (added by the SNDJRNE command or QJOSJRNE API) The journal codes are described in more detail in Journal code descriptions.
27	Journal entry type (JOENTT, *Entry_Type*)	Char (2)	Further identifies the type of user-created or system-created entry. See the Journal code finder for descriptions of the entry types.
29	Time stamp (JOTSTP)	Char (26)	Corresponds to the system date and time when the journal entry was added in the journal receiver. The time stamp is in SAA format. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.

Offset	Field	Format	Description
55	Job name (JOJOB, *Job_Name*)	Char (10)	Specifies the name of the job that added the entry. **Notes:** 1. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the job name. 2. If the job name was not available when the journal entry was deposited, then *NONE is written for the job name.
65	User name (JOUSER, *User_Name*)	Char (10)	Specifies the user profile name of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then blanks are written for the user name.
75	Job number (JONBR, *Job_Number*)	Zoned (6, 0)	Specifies the job number of the user that started the job. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then zeroes are written for the job number.
81	Program name (JOPGM, *Program_Name*)	Char (10)	Specifies 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. **Notes:** 1. If the program that deposited the journal entry is an original program model program, this data will be complete. Otherwise, this data will be unpredictable. 2. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, *OMITTED is given for the program name.
91	Program library name (JOPGMLIB)	Char (10)	The name of the library that contains the program that added the library. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED will be returned for the program library name. IF *NONE is returned for Program name, then *NONE is also returned for the program library name.

Offset	Field	Format	Description
101	Program library ASP device name (JOPGMDEV)	Char (10)	The name of the ASP device that contains the program. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED will be returned for the program library ASP device name. IF *NONE is returned for Program name, then *NONE is also returned for the program library ASP device name.
111	Program library ASP number (JOPGMASP)	Zoned (5,0)	The number for the auxiliary storage pool that contains the program that added the journal entry. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then hexadecimal 0 will be returned for the program library ASP number.
116	Object name (JOOBJ, *Object*)	Char (10)	Specifies the name of the object for which the journal entry was added.[1] This is blank for some entries. If the journaled object is an integrated file system object, then this field is the first 10 bytes of the file identifier.
126	Object library (JOLIB)	Char (10)	Specifies the name of the library containing the object[1]. If the journaled object is an integrated file system object, then the first 6 characters of this field are the last 6 bytes of the file identifier.
136	Member name (JOMBR)	Char (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
146	Count or relative record number (JOCTRR, *Count_Rel_Rec_Num*)	Char (20)	Contains either the relative record number (RRN) of the record that caused the journal entry or a count that is pertinent to type of journal entry.
166	Indicator flag (JOFLAG, *Indicator_Flag*)	Char (1)	Contains an indicator for the operation. The following tables show specific values for this field, if applicable: • APYJRNCHG and RMVJRNCHG journal entries • COMMIT journal entry • INZPFM journal entry • IPL and in-use journal entries • Journal code R (all journal entry types except IL) • ROLLBACK journal entry • Start-journal journal entries
167	Commit control ID (JOCCID, *Commit_Cycle_Id*)	Char (20)	Contains a number that identifies the commit cycle. 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.

Offset	Field	Format	Description
187	User profile (JOUSPF)	Char (10)	Specifies the name of the user profile under which the job was running when the entry was created. **Note:** If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then *OMITTED is given for the user profile.
197	System name (JOSYNM)	Char (8)	Specifies the name of the system on which the entry is being displayed, printed, retrieved, or received 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.
205	Journal identifier (JOJID)	Char (10)	Specifies 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. However, if journaling is stopped, there is no guarantee that the JID will be the same if journaling is started again for the same object. If no JID is associated with the entry, this field has hexadecimal zeros.
215	Referential constraint (JORCST)	Char (1)	Indicates whether this entry was recorded for actions that occurred on records that are part of a referential constraint. The possible values are: **0 =** This entry was not created as part of a referential constraint. **1 =** This entry was created as part of a referential constraint.
216	Trigger (JOTGR)	Char (1)	Indicates whether this entry was created as result of a trigger program. The possible values are: **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.

Offset	Field	Format	Description
217	Incomplete data (JOINCDAT)	Char (1)	Indicates whether this entry has data that is not being retrieved for one of the following reasons: • The length of the entry-specific data exceeds 32 766 bytes. • The entry is associated with a database file that has one or more fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object). The possible values are: **0 =** This entry has all possible data **1 =** This entry has incomplete data. Any data which is marked as incomplete, can only be viewed by using either the QjoRetrieveJournalEntries API, or the command RCVJRNE with any of the following parameters: • ENTFMT(*TYPEPTR) • ENTFMT(*JRNENTFMT) • RTNPTR (with any value specified other than *NONE)
218	Ignored by APYJRNCHG or RMVJRNCHG (JOIGNAPY)	Char (1)	Indicates whether this journal entry will be ignored by the execution of the APYJRNCHG or RMVJRNCHG commands, even though normally this journal entry type has an effect during those command invocations. The possible values are: **0 =** This entry is not ignored by the APYJRNCHG or RMVJRNCHG commands. **1 =** This entry is ignored by the APYJRNCHG or RMVJRNCHG commands.
219	Minimized entry-specific data (JOMINESD)	Char (1)	Indicates whether this entry has minimized entry specific data. The possible values are: **0 =** This entry has complete entry specific data. **1 =** This entry has minimized entry specific data.

Offset	Field	Format	Description
220	Object indicator (JOOBJIND)	Char (1)	An indicator with respect to the information in the object field[2]. 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. **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.
221	System sequence number (JOSYSSEQ)	Char (20)	The system sequence number indicates the relative sequence of when this journal entry was deposited into the journal. You can use the sequence number to sequentially order journal entries that are in separate journal receivers. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then hexadecimal 0 will be returned for the system sequence number.
241	Receiver name (JORCV)	Char (10)	The name assigned to the journal receiver
251	Receiver library name (JORCVLIB)	Char (10)	The name of the library in which the journal receiver resides.
261	Receiver library ASP device name (JORCVDEV)	Char (10)	The name of the ASP device for journal receivers that reside on an independent disk pool
271	Receiver library ASP number (JORCVASP)	Zoned (5,0)	The number of the ASP on which the journal receiver resides.
276	Arm number (JOARM)	Zoned (5,0)	The number of the disk arm that contains the journal entry.
281	Thread identifier (JOTHDX)	Hexadecimal (8)	Identifies the thread within the process that added the journal entry. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then hexadecimal 0 will be returned for the thread identifier.
289	Thread identifier formatted (JOTHD)	Char (16)	See Thread identifier.

Offset	Field	Format	Description
305	Address family (JOADF)	Char (1)	The address family identifies the format of the remote address for this journal entry. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then 0 will be returned for the address family. The possible values are: **0** = This entry was not associated with any remote address. **4** = The format of the remote address is internet protocol version 4.
306	Remote port (JORPORT)	Zoned (5, 0)	The remote port of a the journal entries. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then hexadecimal 0 will be returned for the remote port.
311	Remote address (JORADR)	Char (46)	The remote address of a the journal entries. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then hexadecimal 0 will be returned for the remote address.
357	Logical unit of work (JOLUW)	Char (39)	The logical unit of work identifies entries to be associated with a given unit of work, usually within a commit cycle. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then blanks will be returned for the logical unit of work.
396	Transaction identifier (JOXID)	Char (140)	See the QSYSINC/H.XA header file for the layout of this data. If a RCVSIZOPT or a FIXLENDTA option was specified that omitted the collection of this information, then the displacement to the transaction identifier is 0 and no transaction identifier is returned.
536	Reserved (JORES)	Char (20)	Reserved area. It always contains hexadecimal zeros.

Notes:

[1] If the journal receiver was attached prior to installing V4R2M0 on your system, then the following items are true:

- If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, 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 parameter, the current fully qualified name of the file appears in the converted 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.

[2] 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.

Variable-length portion of the journal entry

For output formats *TYPE1 and *TYPE2, the variable length portion of the journal entry includes just the **Entry-specific data** field. The contents of the Entry-specific data field depends on the journal entry code and entry type. For the layout of the output format *TYPEPTR or *JRNENTFMT, see the QjoRetrieveJournalEntries API. For all other output formats, the variable-length portion of the converted journal entry potentially has two fields:

- Null value indicators
- Entry-specific data

The Null Value Indicators field, contains relevant information only for entries with journal code R. Null value indicators are present in journal entries for record level operations as follows:

- The corresponding physical file has null capable fields.
- The record image has been minimized in the entry specific data.

Otherwise, it contains blanks. If the record image has not been minimized in the entry specific data, the Null Value Indicators field is a character string with one character for each field in the physical file that has record images appearing in the journal. Each character has the following interpretation:

- 0 = corresponding field in the record is not NULL.
- 1 = corresponding field in the record is NULL.

System-supplied output files

The following system-supplied output files define the Null Value Indicators and Entry-Specific Data fields as variable-length character fields:

- QSYS/QADSPJR3
- QSYS/QADSPJR4
- QSYS/QADSPJR5

For additional details regarding the *TYPE3, *TYPE4, and *TYPE5 formats and the exact layout of these two fields, see the following commands:

- Display Journal (DSPJRN)
- Receive Journal Entry (RCVJRNE)
- Retrieve journal entry (RTVJRNE)

Layouts for journal entry types

Use the Journal entry information finder to find the layout for the variable-length portion of the journal entry. Some one journal entry types are described in other places than this topic. The Journal entry information finder indicates those journal entries.

Some journal entry types are documented in QSYSINC library includes, as indicated in the Journal code finder. Some entry types do not have entry-specific data.

These layouts include specific values for fields in the fixed-length portion of the entry and the fields in the entry-specific portion of the entry. The offsets show the relative offset within the Entry-specific data field. The beginning position of the Entry-specific data field depends on the format type that you specify. You can also use the Journal entry information finder to see these layouts.

Layouts for variable-length portion of journal entries

The following tables contain the variable-length portion of the layouts for journal entries.

» Allow use with partial transactions (F MO) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Reason code	Char (1)	**01 =** Partial transactions exist due to restore **02 =** Partial transactions exist because a rollback was ended early.
2	Reserved	Char (3)	Reserved. Set to zeros.
5	Number commit IDs	Bin (32)	The number of commit identifiers.
9	Reserved	Char (72)	Reserved. Set to zeros.
81	Commit Ids	Bin (64) [*]	The array of commit cycle identifiers for partial transactions that remain in the object if the reason code is 01. If the reason code is 02, then an array of Commit Ids is not provided. If the incomplete data indication is ’yes’ when the journal entry is returned on an interface to retrieve journal entries, then this entry is too big to completely return. If pointers were requested on the retrieve interface, then there is a pointer to an array of Commit Ids at this offset. If pointers were not requested on the retrieve interface, then ’*POINTER’ appears at this offset. «

APYJRNCHG (B AT, D DD, E EQ, F AY) and RMVJRNCHG (E EX, F RC) journal entries

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Count or Relative Record Number (JOCTRR)	Zoned (10,0)	Contains the number of journal entries applied or removed. For *TYPE5 output files, the format of this field is Char (20).
	Flag (JOFLAG)	Char (1)	The results of the apply or remove operation: **0 =** Command completed normally. **1 =** Command completed abnormally.
Entry-specific data. This data appears as one field in the standard output formats:			
» 1	First entry applied or removed	Zoned (10,0)	The sequence number of the first entry actually applied or removed. This field is set to -1 if the actual value is larger than 9 999 999 999. See the First entry applied or removed—large field for the actual value.
11	Last entry applied or removed.	Zoned (10,0)	The sequence number of the last entry actually applied or removed. This field is set to -1 if the actual value is larger than 9 999 999 999. See the Last entry applied or removed—large field for the actual value. «
21	Starting receiver name	Char (10)	The name of the first receiver from which entries were applied or removed.
31	Library name	Char (10)	The name of the library for the starting journal receiver.
41	Ending receiver name	Char (10)	The name of the last or ending receiver from which entries were applied or removed.
» 51	Library name	Char (10)	The library for the ending journal receiver.

Relative offset	Field	Format	Description
61	Starting sequence number	Char (10)	The specified starting sequence number for the apply or remove operation. This field is set to -1 if the actual value is larger than 9 999 999 999. See the Starting sequence number—large field for the actual value.
71	Ending sequence number	Char (10)	The specified ending sequence number for the apply or remove operation. This field is set to -1 if the actual value is larger than 9 999 999 999. See the Ending sequence number—large field for the actual value. «
81	Incomplete commit transaction not applied or removed	Char (1)	**0 =** Indicates that either CMTBDY(*NO) was specified or CMTBDY(*YES) was specified and no partial commitment control transactions were found in the range specified by the starting and ending sequence numbers **1 =** Indicates that CMTBDY(*YES) was specified and one or more partial commitment control transactions were found in the range specified by the starting and ending sequence numbers
» 82	First entry applied or removed—large	Char (20)	The sequence number of the first entry actually applied or removed. This field always contains a sequence number.
102	Last entry applied or removed—large	Char (20)	The sequence number of the last entry actually applied or removed. This field always contains a sequence number.
122	Starting sequence number—large	Char (20)	The specified starting sequence number for the apply or remove operation. This field always contains a sequence number.
142	Ending sequence number—large	Char (20)	The specified ending sequence number for the apply or remove operation. This field always contains a sequence number.
162	Number of entries	Char (20)	The number of entries that were applied or removed.
182	Partial transaction starting sequence number	Char (20)	Starting sequence number for any partial transactions that were removed. For integrated file system objects and data areas, this field is always zero.
202	Partial transaction ending sequence number	Char (20)	Ending sequence number for any partial transactions that were removed. For integrated file system objects and data areas, this field is always zero.
222	Number of partial transaction removed	Char (20)	Count of number of entries removed for partial transactions. For integrated file system objects and data areas, this number is always zero.
242	Object deleted	Char (1)	Indicates that the object was deleted during the apply or remove operation. **Y =** Yes **N =** No
243	Object created	Char (1)	Indicates that the object was created during the apply operation. **Y =** Yes **N =** No

Relative offset	Field	Format	Description
244	Early end	Char (1)	Indicates if the apply or remove operation ended early for this object. **Y =** Yes **N =** No
245	Change not made	Char (1)	Indicates that a change was found for this object after an early end to the apply operation. **Y =** Yes **N =** No
246	End reason code	Char (1)	Reason code for early end. See message MCH4801 for the possible values.
247	End message ID	Char (7)	The message identifier associated with an early end to the apply operation
254	Error condition	Bin (31)	The error condition code associated with an early end to the apply operation
258	Partial transactions remain	Char (1)	Indicates that partial transactions remain for this object. **Y =** Yes **N =** No
259	Partial transactions removed	Char (1)	Indicates that at least some partial transactions were removed during the apply operation. **Y =** Yes **N =** No «

Change end of data (F CE) journal entry

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Count or relative record number (JOCTRR)	Zoned (10,0)	The relative record number of the last record kept in the physical file member.

Change journaled object attributes (B JA, D DJ, E EK, F JC) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Attribute changed	Char (1)	Identifies which journal attribute was changed: **1 =** IMAGES **2 =** OMTJRNE **3 =** INHERIT

Relative offset	Field	Format	Description
2	New attributes value	Char (10)	The new value for the attribute that changed. The valid values for each attribute are as follows: • IMAGES(*BOTH) • IMAGES(*AFTER) • OMTJRNE(*NONE) • OMTJRNE(*OPNCLOSYN) • INHERIT(*YES) • INHERIT(*NO) **Note:** Only the characters in the parenthesis appear in this field.

CHGJRN (J NR, J PR) journal entries

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Count or relative record number (JOCTRR)	Zoned (10,0)	Contains the number of receivers attached or detached.
Entry-specific data. This data appears as one field in the standard output formats:			
1	First receiver name	Char (10)	The name of the first receiver that is attached or detached.
11	First receiver library name	Char (10)	The name of the library for the first receiver that is attached or detached.
21	Dual receiver name	Char (10)	The name of the dual receiver that is attached or detached. Blank if only one receiver is used for the journal.
31	Dual receiver library name	Char (10)	The name of the library for the dual receiver that is attached or detached. Blank if only one receiver is used for the journal.

COMMIT (C CM) journal entry

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Count or relative record number (JOCTRR)	Zoned (10,0)	Contains the length of the commit identification.
	Flag (JOFLAG)	Char (1)	Whether the commit operation was initiated by the system or the user: **0 =** All record-level changes were committed for a commit operation initiated by a user. **2 =** All record-level changes were committed for a commit operation initiated by the operating system.
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Commit ID	Char (*)	Contains the commit identification specified by the operation. The Count field specifies the length of this field.

Data queue cleared, has key (Q QJ) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Reserved	Char (2)	Reserved for future use.
3	Key length	Bin (16)	The number of characters in the key.
5	Key order	Char (2)	The Key order is as follows: **GT =** Greater than **LT =** Less than **NE =** Not equal **EQ =** Equal **GE =** Greater than or equal **LE =** Less than or equal
7	Key	Char (*)	The data to be used to remove a message from the data queue.

Delete access path (F PD) journal entry

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Journal identifier (JOJID)	Char (10)	The JID is not provided with the *TYPE1, *TYPE2, and *TYPE3 formats. It can be used with the QJORJIDI API.

Delete receiver (J RD, J RF) journal entries

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Journal identifier (JOJID)	Char (10)	The JID is not provided with the *TYPE1, *TYPE2, and *TYPE3 formats. It can be used with the QJORJIDI API.

Database file OPEN (F OP) and database file CLOSE (F CL) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats.			
1	File Name	Char (10)	The name of the file that was opened or closed. If a physical file is opened, this field and the JOOBJ field are the same. If a logical file is opened, this field contains the name of the logical file. JOOBJ field contains the name of the physical file.
11	Library Name	Char (10)	The library containing the file.
21	Member Name	Char (10)	The file member that was opened of closed.
31	Open options	Char (4)	Only used for file open (entry type OP). Values of the bytes follow:
31	Input	Char (1)	Whether the file was opened for input: **I =** File opened for input **blank=** Input not specified
32	Output	Char (1)	Whether the file was opened for output: **O =** File opened for output **blank=** Output not specified
33	Update	Char (1)	Whether the file was opened for update: **U =** File opened for update **blank=** Update not specified
34	Delete	Char (1)	Indicates if the file was opened for delete: **D =** File opened for delete **blank=** Delete not specified

Force data to auxiliary storage (F FD) journal entry

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Job name (JOJOB)	Char (10)	Blank if the entry is written during IPL or vary on of an independent disk pool.
	Job number (JONBR)	Zoned (6,0)	Zero if entry is written during IPL or vary on of an independent disk pool.
	Program name (JOPGM)	Char (10)	Blank if the entry is written during IPL or vary on of an independent disk pool.

»

Integrated file system begin create (B B0) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Object name offset	Bin (32)	The offset from the beginning of the entry-specific data to the beginning of the object name field.
5	Object type	Char (7)	The object type that was created.
12	Start journaling indicator	Char (1)	Indicates whether journaling will be started. **Y =** Journaling will be started **blank=** Journaling will not be started
13	Reserved	Bin (32)	Reserved. This field is set to zero.
17	Object name	Char (*)	See the Object name for the layout of this field. «

Integrated file system bytes cleared, after-image (B B6) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object.
17	Data length	Bin (64)	The length of the data.
25	Offset	Bin (64)	The offset to begin write of hex zeros (clear).
33	Reserved	Char (16)	Reserved. Set to zeros. «

» Integrated file system change audit attribute (B AA) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Entry type	Char (1)	The type of entry is as follows: **D =** Changed DLO authority **O =** Changed object authority
2	Object name	Char (10)	The name of the object for which the auditing attributes were changed. *N if the object is not in a library.
12	Library name	Char (10)	The name of the library for the object. *N if the object is not in a library.
22	Object type	Char (8)	The type of object.
30	Auditing value	Char (10)	The new value specified on the CHGOBJAUD command.
40	Reserved	Char (135)	Reserved. This field is set to blanks.
175	Object name CCSID	Bin (31)	The coded character set identifier (CCSID) for the object name.
179	Reserved	Char (8)	Reserved. This field is set to blanks.
187	Parent FID	Char (16)	The file identifier of the parent directory
203	Object FID	Char (16)	The file identifier of the object. «

Integrated file system change object authority (B OA) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Entry type	Char (1)	The type of entry. **A** = Change authorization.
2	Object name	Char (10)	The object name. *N if the object is not in a library.
12	Library name	Char (10)	The library name. *N if the object is not in a library.
22	Object type	Char (8)	The type of object.
30	User name	Char (10)	The name of the user profile whose authorization is being granted or revoked.
40	Authorization list name	Char (10)	The name of the authorization list.
50	Object existence authority	Char (1)	**Y =** User has *OBJEXIST authority to the object. **blank=** User does not have *OBJEXIST authority to the object.
51	Object management authority	Char (1)	**Y =** User has *OBJMGT authority to the object **blank=** User does not have *OBJMGT authority to the object.
52	Object operational authority	Char (1)	**Y =** User has *OBJOPR authority to the object. **blank=** User does not have *OBJOPR authority to the object.
53	Authorization list management	Char (1)	Blank if user does not authorization list management to the object
54	Authorization list *PUBLIC authority	Char (1)	**Y =** User has authorization list *PUBLIC authority to the object. **blank=** User does not have authorization list *PUBLIC authority to the object.
55	Read authority	Char (1)	**Y =** User has *READ authority to the object. **blank=** User does not have *READ authority to the object.
56	Add authority	Char (1)	**Y =** User has *ADD authority to the object. **blank=** User does not have *ADD authority to the object.
57	Update authority	Char (1)	**Y =** User has *UPD authority to the object. **blank=** User does not have *UPD authority to the object.
58	Delete	Char (1)	**Y =** User has *DLT authority to the object. **blank=** User does not have *DLT authority to the object.
59	Exclude	Char (1)	**Y =** User has *EXCLUDE authority to the object. **blank=** User does not have *EXCLUDE authority to the object.

Relative offset	Field	Format	Description
60	Execute	Char (1)	**Y =** User has *EXECUTE authority to the object. **blank=** User does not have *EXECUTE authority to the object.
61	Object Alter authority	Char (1)	**Y** = User has *OBJALTER authority to the object. blank= User does not have *OBJALTER authority to the object.
62	Object reference	Char (1)	**Y =** User has *OBJREF authority to the object. **blank =** User does not have *OBJREF authority to the object.
63	Reserved	Char (4)	Reserved. Set to blanks.
67	Operation type	Char (3)	Possible values are: **GRT** = Grant **RPL** = Grant with replace **RVK** = Revoke
70	Reserved	Char (149)	Reserved. Set to blanks.
219	Object name CCSID	Bin (31)	The coded character set identifier (CCSID) for the object name.
223	Reserved	Char (8)	Reserved. Set to blanks.
231	Parent FID	Char (16)	The file identifier of the parent directory.
247	Object FID	Char (16)	The file identifier of the object. «

» Integrated file system change object owner (B OO) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Entry type	Char (1)	The type of entry. **A** = Change owner
2	Object name	Char (10)	The object name. *N if object is not in a library.
12	Library name	Char (10)	The library name. *N if object is not in a library.
22	Object type	Char (8)	The object type.
30	Old owner	Char (10)	The old owner.
40	New owner	Char (10)	The new owner.
50	Reserved	Char (143)	Reserved. Set to blanks.
193	Object name CCSID	Bin (31)	The coded character set identifier (CCSID) for the object name.
197	Reserved	Char (8)	Reserved. Set to blanks.
205	Parent FID	Char (16)	The file identifier of the parent directory.
221	Object FID	Char (16)	The file identifier of the object. «

» Integrated file system change primary group (B OG) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Entry type	Char (1)	Type of entry **A** = Change primary group profile
2	Object name	Char (10)	The object name. *N if object is not in a library.
12	Library name	Char (10)	The library name. *N if object is not in a library.
22	Object type	Char (8)	The type of object.
30	Old PGP	Char (10)	The old primary group.
40	New PGP	Char (10)	The new primary group.
50	Object existence authority	Char (1)	**Y =** New primary group has *OBJEXIST authority to the object. **blank=** new primary group does not have *OBJEXIST authority to the object.
51	Object management authority	Char (1)	**Y =** New primary group has *OBJMGT authority to the object. **blank=** New primary group does not have *OBJMGT authority to the object.
52	Object operational authority	Char (1)	**Y =** New primary group has *OBJOPR authority to the object. **blank=** New primary group does not have *OBJOPR authority to the object.
53	Object Alter authority	Char (1)	**Y =** New primary group has *OBJALTER authority to the object. **blank=** New primary group does not have *OBJALTER authority to the object.
54	Object reference	Char (1)	**Y =** New primary group has *OBJREF authority to the object. **blank=** New primary group does not have *OBJREF authority to the object.
55	Reserved	Char (10)	Reserved. Set to blanks.
65	Authorization list management	Char (1)	Blank if new primary group does not authorization list management to the object
66	Read authority	Char (1)	**Y =** New primary group has *READ authority to the object. **blank=** New primary group does not have *READ authority to the object.
67	Add authority	Char (1)	**Y** New primary group has *ADD authority to the object. **blank=** New primary group does not have *ADD authority to the object.

Relative offset	Field	Format	Description
68	Update authority	Char (1)	**Y =** New primary group has *UPD authority to the object. **blank=** New primary group does not have *UPD authority to the object.
69	Delete	Char (1)	**Y =** New primary group has *DLT authority to the object. **blank=** New primary group does not have *DLT authority to the object.
70	Execute	Char (1)	**Y =** New primary group has *EXECUTE authority to the object. **blank=** New primary group does not have *EXECUTE authority to the object.
71	Reserved	Char (10)	Reserved. Set to blanks.
81	Exclude	Char (1)	**Y =** New primary group has *EXCLUDE authority to the object. **blank=** New primary group does not have *EXCLUDE authority to the object.
82	Revoke previous primary group authority	Char (1)	**Y =** The previous primary group authority to the object was revoked. **blank =** The previous primary group authority to the object was not revoked.
83	Reserved	Char 143	Reserved. Set to blanks.
226	Object name CCSID	Bin (31)	The coded character set identifier (CCSID) for the object name.
230	Reserved	Char (8)	Reserved. Set to blanks.
238	Parent FID	Char (16)	The file identifier of the parent directory.
254	Object FID	Char (16)	The file identifier of the object. «

» Integrated file system create-summary (B B1) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Offset to name	Bin (32)	The offset from the beginning of the entry-specific data to the beginning of the object name field.
5	Offset to path name	Bin (32)	The offset from the beginning of the entry-specific data to the beginning of the Path name field.
9	Offset to symbolic link contents field.	Bin (32)	The offset from the beginning of the entry-specific data to the beginning of the Symbolic link contents field.
13	Object type	Char (7)	The object type that was created.

Relative offset	Field	Format	Description
20	Scan attribute	Char (1)	If the new object is a stream file (*STMF), this field is the scan (QP0L_ATTR_SCAN) attribute of the Set Attributes (Qp0lSetAttr()) API. If the new object is a directory (*DIR), this field is the create object scanning (QP0L_ATTR_CRTOBJSCAN) attribute of the Qp0lSetAttr()) API. For details on the QP0L-ATTR_SCAN and QP0L_ATTR_CRTOBJSCAN attributes, see the Set Attributes (Qp0lSetAttr()) API.
21	File ID of object	Char (16)	The new object file identifier.
37	Owner name	Char (10)	The user profile name of the owner.
47	Group name	Char (10)	The primary group profile name
57	Auditing value	Char (10)	The auditing value of the new object.
67	Object CCSID	Bin (31)	The coded character set identifier (CCSID) for the object.
71	Object existence authority	Char (1)	**Y =** The owner has *OBJEXIST authority to the object. **blank=** The owner does not have *OBJEXIST authority to the object.
72	Object management authority	Char (1)	**Y =** The owner has *OBJMGT authority to the object **blank=** The owner does not have *OBJMGT authority to the object.
73	Object operational authority	Char (1)	**Y =** The owner has *OBJOPR authority to the object. **blank=** The owner does not have operational authority to the object.
74	Object Alter authority	Char (1)	**Y =** The owner has *OBJALTER authority to the object. **blank =** The owner does not have *OBJALTER authority to the object.
75	Object reference	Char (1)	**Y =** The owner has *OBJREF authority to the object. **blank=** The owner does not have *OBJREF authority to the object.
76	Read authority	Char (1)	**Y =** The owner has *READ authority to the object. **blank=** The owner does not have *READ authority to the object.
77	Add authority	Char (1)	**Y =** The owner has *ADD authority to the object. **blank=** The owner does not have *ADD authority to the object.
78	Update authority	Char (1)	**Y =** The owner has *UPD authority to the object. **blank=** The owner does not have *UPD authority to the object.
79	Delete	Char (1)	**Y =** The owner has *DLT authority to the object. **blank=** The owner does not have *DLT authority to the object.

Relative offset	Field	Format	Description
80	Exclude	Char (1)	**Y =** The owner has *EXCLUDE authority to the object. **blank=** The owner does not have *EXCLUDE authority to the object.
81	Execute	Char (1)	**Y =** The owner has *EXECUTE authority to the object. **blank=** The owner does not have *EXECUTE authority to the object.
82	Reserved	Char (1)	Reserved. Set to blank.
83	Object existence authority	Char (1)	**Y =** The primary group has *OBJEXIST authority to the object. **blank=** The primary group does not have *OBJEXIST authority to the object.
84	Object management authority	Char (1)	**Y =** The primary group has *OBJMGT authority to the object **blank=** The primary group does not have *OBJMGT authority to the object.
85	Object operational authority	Char (1)	**Y =** The primary group has *OBJOPR authority to the object. **blank=** The primary group does not have operational authority to the object.
86	Object Alter authority	Char (1)	**Y =** The primary group has *OBJALTER authority to the object. **blank=** The primary group does not have *OBJALTER authority to the object.
87	Object reference	Char (1)	**Y =** The primary group has *OBJREF authority to the object. **blank=** The primary group does not have *OBJREF authority to the object.
88	Read authority	Char (1)	**Y =** The primary group has *READ authority to the object. **blank=** The primary group does not have *READ authority to the object.
89	Add authority	Char (1)	**Y =** The primary group has *ADD authority to the object. **blank=** The primary group does not have *ADD authority to the object.
90	Update authority	Char (1)	**Y =** The primary group has *UPD authority to the object. **blank=** The primary group does not have *UPD authority to the object.
91	Delete	Char (1)	**Y =** The primary group has *DLT authority to the object. **blank=** The primary group does not have *DLT authority to the object.

Relative offset	Field	Format	Description
92	Exclude	Char (1)	**Y =** The primary group has *EXCLUDE authority to the object. **blank=** The primary group does not have *EXCLUDE authority to the object.
93	Execute	Char (1)	**Y =** The primary group has *EXECUTE authority to the object. **blank=** The primary group does not have *EXECUTE authority to the object.
94	Reserved	Char (1)	Reserved. Set to blank.
95	Object existence authority	Char (1)	**Y =** *PUBLIC has *OBJEXIST authority to the object. **blank=** *PUBLIC does not have *OBJEXIST authority to the object. This field is only used when Authorization list *PUBLIC is blank.
96	Object management authority	Char (1)	**Y =** *PUBLIC has *OBJMGT authority to the object **blank=** *PUBLIC does not have *OBJMGT authority to the object. This field is only used when Authorization list *PUBLIC is blank.
97	Object operational authority	Char (1)	**Y =** *PUBLIC has *OBJOPR authority to the object. **blank=** *PUBLIC does not have operational authority to the object. This field is only used when Authorization list *PUBLIC is blank.
98	Object Alter authority	Char (1)	**Y =** *PUBLIC has *OBJALTER authority to the object. **blank=** *PUBLIC does not have *OBJALTER authority to the object. This field is only used when Authorization list *PUBLIC is blank.
99	Object reference	Char (1)	**Y =** *PUBLIC has *OBJREF authority to the object. **blank=** *PUBLIC does not have *OBJREF authority to the object. This field is only used when Authorization list *PUBLIC is blank.
100	Read authority	Char (1)	**Y =** *PUBLIC has *READ authority to the object. **blank=** *PUBLIC does not have *READ authority to the object. This field is only used when Authorization list *PUBLIC is blank.
101	Add authority	Char (1)	**Y =** *PUBLIC has *ADD authority to the object. **blank=** *PUBLIC does not have *ADD authority to the object. This field is only used when Authorization list *PUBLIC is blank.

Relative offset	Field	Format	Description
102	Update authority	Char (1)	**Y =** *PUBLIC has *UPD authority to the object. **blank=** *PUBLIC does not have *UPD authority to the object. This field is only used when Authorization list *PUBLIC is blank.
103	Delete	Char (1)	**Y =** *PUBLIC has *DLT authority to the object. **blank=** *PUBLIC does not have *DLT authority to the object. This field is only used when Authorization list *PUBLIC is blank.
104	Exclude	Char (1)	**Y =** *PUBLIC has *EXCLUDE authority to the object. **blank=** *PUBLIC does not have *EXCLUDE authority to the object. This field is only used when Authorization list *PUBLIC is blank.
105	Execute	Char (1)	**Y =** *PUBLIC has *EXECUTE authority to the object. **blank=** *PUBLIC does not have *EXECUTE authority to the object. This field is only used when Authorization list *PUBLIC is blank.
106	Reserved	Char (1)	Reserved. Set to blank.
107	Authorization list name	Char (10)	The authorization list name for the new object.
117	Authorization list *PUBLIC	Char (1)	The authorization List *PUBLIC authority. Possible values are **Y** or blank.
118	Format indicator.	Char (1)	The format indicator is set to one of the following values: **0 =** The original layout of this journal entry (FORMAT1) **1 =** The layout of FORMAT1 plus the Device id field is set appropriately (FORMAT2) **2 =** The layout for all of FORMAT2 plus the following fields are set appropriately (FORMAT3): • Scan attribute • Create object auditing • S_ISVTX value • S_ISUID value • S_ISGID value For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
119	PC read-only	Char (1)	The PC read Only flag. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
120	PC hidden	Char (1)	The PC hidden flag. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
121	PC system	Char (1)	The PC System file flag. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
122	PC changed	Char (1)	The PC changed flag. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.

Relative offset	Field	Format	Description
123	Journal information	Char (36)	The journaling information for the new object. This field indicates if journaling is now active on the new object. If so, it also contains the information the information used to start journaling. See Journal information for the layout of this field.
159	Device ID	Bin (64)	This field is only valid when the object type is *CHRSF.
167	Create object auditing	Char (10)	The create object auditing value. This value only applies to directories (*DIR).
177	S_ISVTX value	Char (1)	The restricted rename and unlink (S_ISVTX) mode bit. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
178	S_ISUID value	Char (1)	The S_ISUID mode bit. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
179	S_ISGID value	Char (1)	The S_ISGID mode bit. For information about the values in this field see the Get Attributes (Qp0lGetAttr()) API.
180	Object name	Char (*)	See Object name for the layout of this field.
*	Path name	Char (*)	See Path name for the layout of this field.
*	Symbolic link contents	Chare (*)	See Symbolic link contents for the layout of this field.«

» Integrated file system end journaling for object (B ET) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object. «

» Integrated file system link to existing object (B B2) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the existing object.
17	Link offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Link name field.
21	Path offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Path name field.
25	Start journaling	Char (1)	The start journaling indicator. **Y =** Journaling starts on the existing object as a result of this operation. The **Journal information** field has start journaling information. **blank=** Journaling is not started on the existing object as a result of this operation. The Journal information field contains all hex zeros.

Relative offset	Field	Format	Description
26	Summary	Char (1)	The summary record indicator. **Y =** This journal entry was deposited after the actual operation was completed. If the **Start journaling field** is **Y**, then the **Journal information** field contains the actual information related to starting journaling on the target object. If the **Start journaling** field is blank, then the **Journal information** field contains all hex zeros. **blank=** This journal entry was deposited before the actual operation was attempted. If the **Start journaling field** is **Y**, then the **Journal information** field contains the journal information inherited from its new parent. This information is used to attempt a start journaling operation. If the **Start journaling** field is blank, then the **Journal information** field contains all hex zeros.
27	Reserved	Char (2)	Reserved. Set to zero.
29	Journal information	Char (36)	The journaling information for the new object. This field is defined in Journal information.
65	Link name	*	The name of the new link to the object. See Object name for the layout of this field.
Char (*)	Path name	Char (*)	The existing object path name. If this B2 journal entry was deposited as a result of a rollback of a B5 entry, then this will actually be the path to the parent directory to which the link is being added. See Path name for the layout of this field. «

» Integrated file system object attribute changed (B FA) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object.
17	Parent FID	Char (16)	The file identifier of the parent directory.
33	Object type	Char (7)	The type of object.
40	Reserved	Char (9)	Reserved for alignment. This field is set to hex zeros.
49	Next attribute offset	Bin (32)	The offset to the next attribute
53	Attribute identifier	Bin (32)	The attribute identifier. See the Set Attributes (Qp0lSetAttr()) API for information about the structure and content of this field.
57	Reserved	Char (4)	Reserved field
61	Reserved	Char (4)	Reserved field
65	Changed data	Char (*)	The data that was changed. «

» Integrated file system object closed (B CS) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Open flags	Bin (31)	Open flags. See the Open API for a description of these flags. «

» Integrated file system object deleted (B BD) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object. «

» Integrated file system object forced (B FC) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object. «

» Integrated file system object opened (B OF) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Open flags	Bin (31)	Open flags. See the Open API for a description of these flags. «

» Integrated file system object truncated (B TR) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object.
17	Old size	Bin (64)	The size of the object in bytes before it was truncated.
25	New size	Bin (64)	The size of the object in bytes after it was truncated. «

» Integrated file system remove link (link) (B B5) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the existing object.
17	Parent FID	Char (16)	The file identifier of the object parent directory.
33	Link offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the link name field.
37	Parent path offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Parent path field.
41	Parent directory JID	Char (10)	The journal identifier of parent directory.
51	Object type	Char (7)	The type of the object.
58	Reserved	Char (3)	Reserved. Set to zero.
61	Internal data offset	Bin (32)	The offset from beginning of this field to the beginning of Internal data field.
65	Link name	Char (*)	The name of link. See Object name for the layout of this field.
*	Parent path	Char (*)	The path to the parent that used to contain this link. See Path name for the layout of this field.
*	Internal data	Char (*)	Internal data. «

» Integrated file system remove link (parent directory) (B B4) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the existing object.
17	Parent FID	Char (16)	The parent directory of the link file identifier.
33	Link offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Link name field.
37	Object JID	Char (10)	The journal identifier of the object.
47	Object type	Char (7)	The type of the object.
54	Reserved	Char (7)	Reserved. Set to zero
61	System offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of Internal data.
65	Link name	Char (*)	The name of link. See Object name for the layout of this field.
*	Internal data	Char (*)	Internal data. «

» Integrated file system rename file identifier (B RN) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Old FID	Char (16)	The file identifier of the object before the rename operation.
17	Reserved	Char (14)	Reserved. Set to blanks.
31	New FID	Char (16)	The file identifier of the object after the rename operation.
47	Reserved	Char (14)	Reserved. Set to blanks. «

» Integrated file system rename, move object (B B3) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object for the renamed link.
17	Source parent FID	Char (16)	The file identifier of the source object directory.
33	Target parent FID	Char (16)	The file identifier of the target object directory.
49	Replaced object FID	Char (16)	The file identifier of the object that was replaced by this operation. This field contains all hex zeros if no object was replaced.
65	Source offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Source name field.
69	Target offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Target name field.
73	Source parent offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Source parent path field.
77	Target parent offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of the Target parent path field.
81	Start journaling	Char (1)	The start journaling indicator: **Y =** Journaling starts on the existing object as a result of this operation. The **Journal information** field contains the information used to start journaling. **blank=** Journaling does not start on the existing object as a result of this operation. The **Journal information** field contains all hex zeros.

Relative offset	Field	Format	Description
82	Summary	Char (1)	The summary record indicator: **Y =** This journal entry was deposited after the actual operation was completed. If the **Start journaling** field is also **Y**, then the **Journal information** field contains the actual information related to starting journaling on the target object. If the **Start journaling** field is blank, then the Journal information record will contain all hex zeros. **blank=** This journal entry was deposited before the actual operation was attempted. If Start journaling is **Y**, then the **Journal information** field contains the journal information inherited from its new parent. That information is used to attempt a start journaling operation. If the **Start journaling** field is blank, then the **Journal information** field contains all hex zeros.
83	Replace	Char (1)	The replace indicator. Indicates if the target was replaced as a result of this operation. **Y =** Indicates that the target was replaced. **blank=** Indicates that the target did not exist before this operation.
84	Journal entry flags	Bin (32)	The fields for journal entry flags are as follows: **Both journaled** Bit(0)—1 = Indicates that this entry is one of a pair of B3 entries sent for this move operation. This occurs when both the source and target parent directories are journaled at the time of the move operation. **Source entry** Bit(1)—1 = Indicates that this entry was deposited because the source parent was journaled. **Reserved** Bits(2-7)—Reserved. Set to zero.
88	Reserved	Char (4)	Reserved field.
92	Journal information	Char (37)	The journaling information for the new object. This field is defined in Journal information.
129	System offset	Bin (32)	The offset from beginning of this entry-specific data to the beginning of Internal data field.
133	Source name	Char (*)	The name of object being renamed or moved. See Object name for the layout of this field.
*	Target name	Char (*)	The new name of object after being renamed or moved. See Object name for the layout of this field.
*	Source parent path	Char (*)	The path to the parent directory from which the object previously belonged. See Path name for the layout of this field.
*	Target parent path	Char (*)	The path to the parent directory to which the object now belongs. See Path name for the layout of this field.
*	Internal data	Char (*)	Internal data. ≪

Integrated file system storage for object freed (B FF) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object.
17	Old size	Bin (64)	The old size of the object.

Integrated file system write, after-image (B WA) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object FID	Char (16)	The file identifier of the object.
17	Data length	Bin (64)	Length of the data.
25	Offset	Bin (64)	The offset to begin write.
33	Reserved	Char (16)	Reserved field
49	Data	Char (*)	The actual data that was written. If the incomplete data indicator is off, the information is a character string. Otherwise, it is a pointer to the actual data. See Work with pointers in journal entries for more information.

INZPFM (F IZ) journal entry

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Count or Relative Record Number (JOCTRR)	Zoned (10,0)	Contains the number of records specified on the TOTRCDS parameter of the Initialize Physical File Member (INZPFM) command.
	Flag (JOFLAG)	Char (1)	Indicates the type of record initialization that was done: **0 =** *DFT (default) **1 =** *DLT (delete)
Entry-specific data. This data appears as one field in the standard output formats:			
1	Entry-specific data		If the member is initialized with default records, this field contains the default record image.

IPL (J IA, J IN) and in-use (B OI, C BA, D ID, E EI, F IU, I DA, J JI, Q QI) journal entries

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Time stamp (JOTIME)	Zoned (6,0)	The timestamp created at IPL is read from the battery-powered clock. If the battery-powered clock cannot be read, the time is that of the system power down, not the time of the IPL, because the system time has not yet been updated at the time the journal entry is written.
	Flag (JOFLAG)	Char (1)	For in-use entries, indicates whether the object was synchronized with the journal: **0 =** Object was synchronized with journal **1 =** Object was not synchronized with journal.

Journal code R, all journal entry types except IL

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Flag (JOFLAG)	Char (1)	Whether a before-image is present[1]: **0 =** Before-image is not present. If before-images are being journaled, this indicates that an update operation or delete operation is being requested for a record that has already been deleted. **1 =** Before-image is present.
	Journal identifier (JOJID)	Char (10)	The JID is not provided with the *TYPE1, *TYPE2, and *TYPE3 formats. It can be used with the QJORJIDI API.
Entry-specific data. This data appears as one field in the standard output formats:			
1	Entry-specific Data	Char (*)	After-image of the record for entry types PT, PX, UP, or UR. Before-image of the record for entry types UB, DL, BR, or DR if before-images are being journaled and the record was not previously deleted.
Note: [1]The flag does not apply to these entry types: PT, PX, UP, and UR.			

License key not valid (L LK) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Product ID	Char (7)	The ID of the product whose license key was not valid.
8	License Term	Char (6)	The term of the license.
14	Feature	Char (4)	The product feature code.
18	Usage Limit	Zoned (6,0)	The usage limit for the product.
24	License Key	Char (18)	The license key for the product.
42	Expiration Date	Char (7)	The expiration date for the license key.

Relative offset	Field	Format	Description
49	Vendor Data	Char (8)	Data placed in the entry by the product vendor.
57	Processor Group	Char (3)	The processor group for the license key.

Logical unit of work (C LW) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	LUW header portion	416	The header portion of the entry-specific data contains general information about the logical unit of work (LUW). The layout for the Logical Unit of Work journal entry-header record describes the contents of the header portion.
After the header portion	LUW local portion	80	Information about local resources that participated in the LUW. The entry might have 0 to n records for local locations. Each local record is 48 characters long. The layout for the Logical Unit of Work journal entry-local record describes the local record.
After the local portion	LUW API portion	112	Information about API resources that participated in the LUW. The entry might have 0 to n records for API resources. Each API resource record is 80 characters long. The layout for the 0 Logical Unit of Work journal entry-API record describes the API record.
After the API portion	LUW DDL portion	96	Information about DDL resources that participated in the LUW. The entry might have 0 to n records for DDL resources. Each DDL resource record is 80 characters long. The layout for the Logical Unit of Work journal entry-DDL record describes the DDL record.
After the DDL portion	LUW remote portion	128	Information about remote locations that participated in the LUW. The entry might have 0 to n records for remote locations. Each remote location record is 128 characters long. The layout for the Logical Unit of Work journal entry-RMT record describes the remote record.
After the remote portion	LUW DDM portion	96	Information about DDM resources that participated in the LUW. The entry might have 0 to n records for DDM resources. Each DDM resource record is 96 characters long. The layout for the Logical unit of work (C LW) journal entry - DDM record describes the DDM record.

Logical unit of work (C LW) journal entry - API record

Relative offset	Field	Format	Description
1	Record type	Char (4)	Type of record: **API =** API Commitment Resource record
5	Record Length	Bin (15)	Length of record. Currently 80 for API record.

Relative offset	Field	Format	Description
7	Record Position	(4)[1]	This identifies the position in the LUW journal entry where this record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains this record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains this record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where this record starts within this journal entry. This is the number of bytes past the beginning of the entry where this record starts. For example, 0 means the first byte in the entry.
11	Resource Location Position	(4)[1]	This identifies the position in the LUW journal entry where the LCL record starts for this API resource's location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry.
15	Next Resource Position	(4)[1]	This identifies the position in the LUW journal entry where the next API or DDL record starts for this API resource's location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 indicates that this is the last resource for this API resource's location.
19	API Resource	Char (10)	Name of API resource.
29	API Program	Char (20)	Name of the exit program for the API resource: • Char (10): exit program name • Char (10): exit program library
49	Journal	Char (20)	Journal related to the location for this resource: • Char (10): Journal name (blank if this resource belongs to the location with no journal) • Char (10): Journal library (blank if this resource belongs to the location with no journal)

Relative offset	Field	Format	Description
69	Commit Cycle ID	Bin (31)	The commit cycle identifier for the journal. This is 0 if this resource belongs to the location with no journal. This is -1 if the actual commit cycle identifier value is larger than 2 147 483 647. The Commit Cycle ID Long field always contains the correct value.
73	Commit Protocol	Char (1)	The commit protocol for this resource: **2 =** This is a two-phase resource (API resources are always two-phase resources).
74	Resource Usage	Char (2)	The currently allowed access for this resource. The allowed access for some resources can change from one LUW to another depending on whether one-phase resources are registered: **RO =** This resource is currently read-only. Updates were not made during the LUW. **UP =** This resource is currently able to be updated. Updates might or might not have been made during the LUW.
76	API State	Char (2)	Indicates whether the API resource was committed or rolled back successfully: **CS =** This resource was committed successfully. **RS =** This resource was rolled back successfully. **CF =** An attempt to commit this resource failed. **RF =** An attempt to rollback this resource failed.
78	API Last Agent Flag	Char (1)	Whether this resource is to be selected as the last agent during all commit requests: **Y =** This resource is to be selected as the last agent. **N =** This resource is not to be selected as the last agent.
79	Allow Remote Resources	Char (1)	Whether remote resources are allowed to participate in a LUW with this resource: **Y =** Remote resources are allowed with this resource. **N =** Remote resources are not allowed with this resource.
80	Save While Active Flag	Char (1)	Whether this resource will hold out a save-while-active request until a commitment boundary is reached: **Y =** This resource will hold save-while-active requests. **N =** This resource will not hold save-while-active requests.
81	Commit Cycle ID Long	Zoned (20,0)	The commit cycle identifier for the journal. This is 0 if this resource belongs to the location with no journal.
101	Reserved	Char (12)	Reserved for future use.
Note: [1]The format for this field is in the description.			

Logical unit of work (C LW) journal entry - DDL record

Relative offset	Field	Format	Description
1	Record Type	Char (4)	Type of record: **DDL =** SQL Object Change record.
5	Record Length	Bin (15)	Length of record. Currently 624 for DDL record.
7	Record Position	(4)[1]	This identifies the position in the LUW journal entry where this record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains this record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains this record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where this record starts within this journal entry. This is the number of bytes past the beginning of the entry where this record starts. For example, 0 means the first byte in the entry.
11	Resource Location Position	(4)[1]	This identifies the position in the LUW journal entry where the LCL record starts for this DDL resource's location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry.
15	Next Resource Position	(4)[1]	This identifies the position in the LUW journal entry where the next API or DDL record starts for this DDL resource's location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 indicates that this is the last resource for this DDL resource's location.

Relative offset	Field	Format	Description
19	DDL Resource Information	Char (29)	Object identification and operation performed on object: - Char (10): First 10 characters of object name. The object name field always contains the full object name. - Char (10): Object library name - Char (7): Object type (*FILE, *LIB or *SQLPKG) - Char (2): Object operation The possible object operations and their meanings are the following: **AC** = Add PF Constraint **CC** = Create Collection **CF** = Create File **CG** = Create Program **CM** = Create Member **CP** = Create SQL Package **CS** = Create Service Program **CT** = Create User Defined Type **DC** = Delete Collection **DF** = Delete File **DG** = Drop Program **DP** = Delete SQL Package **DS** = Drop Service Program **DT** = Drop User Defined Type **FC** = Change File **FR** = Rename File **GF** = Grant Files **GG** = Grant Program **GP** = Grant to SQL Package **GR** = Grant Java(TM) Routine **GS** = Grant Service Program **GT** = Grant User Defined Type **OP** = COMMENT ON SQL Package **OT** = COMMENT User Defined Type **RC** = Remove PF Constraint **RG** = Revoke Program **RF** = Revoke Files **RP** = Revoke from SQL Package **RR** = Revoke Java Routine **RS** = Revoke Service Program **RT** = Revoke User Defined Type **TA** = Add PF Trigger **TR** = Remove PF Trigger **UL** = Unlink Datalink **XF** = Transfer Files
48	Reserved	Char (1)	Reserved for future use.
49	Journal	Char (20)	Journal related to the location for this resource: - Char (10): Journal name (blank if this resource belongs to the location with no journal) - Char (10): Journal library (blank if this resource belongs to the location with no journal)
69	Commit Cycle ID	Bin (31)	The commit cycle identifier for the journal. This is 0 if this resource belongs to the location with no journal. This is -1 if the actual commit cycle identifier value is larger than 2 147 483 647. The Commit Cycle ID Long field always contains the correct value.

Relative offset	Field	Format	Description
73	Commit Protocol	Char (1)	The commit protocol for this resource: **2 =** This is a two-phase resource (DDL resources are always two-phase resources).
74	DDL State	Char (2)	Indicates whether the DDL resource was committed or rolled back successfully: **CS =** This resource was committed successfully. **RS =** This resource was rolled back successfully. **CF =** An attempt to commit this resource failed. **RF =** An attempt to rollback this resource failed.
76	Commit Cycle ID Long	Zoned (20,0)	The commit cycle identifier for the journal. This is 0 if this resource belongs to the location with no journal.
96	Object Name	Char (288)	The full object name
384	Reserved	Char (1)	Reserved for future use.
Note: [1]The format for this field is in the description.			

Logical unit of work (C LW) journal entry - DDM record

Relative offset	Field	Format	Description
1	Record Type	Char (4)	Type of record: **DDM =** Remote Database File record.
5	Record Length	Bin (15)	Length of record. Currently 96 for DDM record.
7	Record Position	(4)[1]	This identifies the position in the LUW journal entry where this record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains this record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains this record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where this record starts within this journal entry. This is the number of bytes past the beginning of the entry where this record starts. For example, 0 means the first byte in the entry.

Relative offset	Field	Format	Description
11	Resource Location Position	(4)[1]	This identifies the position in the LUW journal entry where the RMT record starts for this DDM file's location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry.
15	Next Resource Position	(4)[1]	This identifies the position in the LUW journal entry where the next DDM record starts for this DDM file's location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 indicates that this is the last resource for this DDM file's location.
19	DDM File	Char (20)	Name of the DDM file and library for the open remote file: • Char (10): DDM file name • Char (10): DDM file library name
29	Remote Location Information	Char (54)	Identification of the remote location and communication information for this resource's location: • Char (10): Remote Location name • Char (10): Device name • Char (10): Mode • Char (8): Remote network ID • Char (8): Conversation correlator network ID • Char (8): Transaction program name
93	Open Flag	Char (1)	Whether the DDM file was open or closed when this LUW ended: **O =** The DDM file was open. **C =** The DDM file was closed.
94	Commit Protocol	Char (1)	The commit protocol for this resource: **1 =** This is a one-phase resource. **2 =** This is a two-phase resource.

Relative offset	Field	Format	Description
95	Resource Usage	Char (2)	The currently allowed access for this resource. The allowed access for some resources can change from one LUW to another depending on whether one-phase resources are registered: **RO =** This resource is currently read-only. Updates were not made during the LUW. **UP =** This resource is currently able to be updated. Updates might or might not have been made during the LUW. **Note:** This does not indicate whether updates were actually made during the LUW. It only indicates whether updates are allowed, given the other resources currently registered.
Note: [1]The format for this field is in the description.			

Logical unit of work (C LW) journal entry-header record

Relative offset	Field	Format	Description
1	Record type	Char (4)	Type of record: **HDR =** Header record.
5	Record length	Bin (15)	Length of record. Currently 400 for HDR record.
7	Record position	(4)[1]	This identifies the position in the LUW journal entry where this record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains this record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains this record (1 for the first, 2 for the second, and so forth). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where this record starts within this journal entry. This is the number of bytes past the beginning of the entry where this record starts. For example, 0 means the first byte in the entry. Because they always start at the beginning of the journal entry, this offset is always 0 for HDR records.
11	Number of journal entries	Bin (15)	The number of actual journal entries sent for this LUW journal entry. This is 1 unless the LUW journal entry is greater than 32K-1 bytes.

Relative offset	Field	Format	Description
13	Location with no journal position	(4)[1]	This identifies the position in the LUW journal entry where the LCL record starts for the local location with no journal. It is made up of two numbers: - Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so forth). Note that this is **not** the actual journal entry sequence number. - Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 means that there is no local location that does not have a journal.
17	First location with journal position	(4)[1]	This identifies the position in the LUW journal entry where the LCL record starts for the first local location with a journal. It is made up of two numbers: - Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. - Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 means that there are no local locations with a journal.
21	First remote location position	(4)[1]	This identifies the position in the LUW journal entry where the RMT record starts for the first remote location. It is made up of two numbers: - Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. - Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 means there are no remote locations.

Relative offset	Field	Format	Description
25	LUW operation	Char (2)	The operation that was performed to end this LUW: **CM =** A commit operation was performed. This does not necessarily mean that the resources were committed. In some cases a commit operation is changed to a rollback operation with respect to two-phase commit rules. **RB =** A rollback operation was performed. An attempt was made to roll back all resources.
27	Protected logical unit of work identifier (LUWID)	Char (41)	The format for the LUWID is: • Bin (15): The total length of the LUWID not including this field • Char (0 to 8): The network ID • Char (1): The separator character **.** • Char (0 to 8): The local location name • Char (3): The separator characters **.X′** • Char (12): The hex value of the instance number converted to character • Char (2): The separator characters **′.** • Char (5): The hex value of the sequence number converted to decimal
68	Unprotected logical unit of work identifier	Char (41)	The format for the LUWID for unprotected conversations is the same as for protected conversations.
109	Default Journal Commit Cycle ID	Bin (31)	The commit cycle identifier for the default journal for this LUW. This is 0 if no commit cycle was started for this journal during this LUW. This is -1 if the actual commit cycle identifier value is larger than 2 147 483 647. The Default Journal Commit Cycle ID Long field always contains the correct value.
113	Commitment definition name	Char (10)	The name of the commitment definition for which this LUW took place.
123	Commitment definition identifier	Char (10)	The commitment definition identifier of the commitment definition. This is not useful to the user.
133	Qualified job name	Char (26)	The job that created the commitment definition.
159	Reserved	Char (1)	Reserved for future use. Currently always blank.
160	Commitment definition scope	Char (1)	The scope of the commitment definition: **A =** Activation group level commitment definition. **E =** Explicitly named commitment definition. **J =** JOB commitment definition.
161	Activation group mark	Bin (31)	The activation group mark for the commitment definition: **0 =** This is the *JOB or an explicitly named commitment definition. **2 =** This is the *DFTACTGRP commitment definition. **# =** The number of the activation group for this activation group level commitment definition.

Relative offset	Field	Format	Description
165	Notify object	Char (37)	The notify object for the commitment definition: • Char (10) - Object name • Char (10) - Object library • Char (10) - Object member (blank if object is not a file) • Char (7) - Object type (*MSGQ, *DTAARA or *FILE)
202	Default journal	Char (20)	The default journal for the commitment definition: • Char (10): Journal name • Char (10): Journal library
222	Initiation type	Char (1)	Whether this commit or rollback operation was initiated by the user or by the system: **E =** Explicit commit or rollback operation initiated by the user. **I =** Implicit commit or rollback operation due to activation group end, job end, or system end. If the LUW was finished after a system end, this is set to **I**, even if an explicit commit or rollback operation was running at the time the system ended.
223	LUW end status	Char (1)	Indication of when this LUW ended with respect to the job that created the commitment definition for which this LUW took place: **N =** The LUW ended while the job was running normally. **E =** The LUW ended during job end. This means that the LUW was still pending when a request was made to end the job. If the requested operation is **CM**, then a commit request had started before the request to end the job and was finished during the job-end phase. **I =** The LUW ended during the IPL following a system end. If the requested operation is **CM**, then a commit request was started before the system end and was finished during the IPL. **P =** The LUW ended after the IPL following a system end. In this case, the requested operation is **CM** and the LUW was prepared pending the commit/rollback decision from the initiator or last agent when the system ended. During the IPL, local resources were brought back to a prepared state in a system database server job. After resynchronization was performed to learn the commit/rollback decision, the LUW ended by committing or rolling back the local resources in that same system database server job.

Relative offset	Field	Format	Description
224	Sync-point Role	Char (1)	The sync-point role played by this location during a commit operation: **I =** Initiator: the root of the sync-point tree. **C =** Cascaded initiator: an intermediate location in the sync-point tree. **A =** Agent: a leaf location in the sync-point tree.C tree. **blank=** This LUW ended in a rollback request.
225	Partner role	Char (1)	The partner role played by this location during a commit: **I =** Initiator: the root of the sync-point tree. **N =** Not-last agent: a prepare request was sent to this location during the prepare wave. **L =** Last agent: a prepare request was not sent to this location during the prepare wave. Instead, a request was made to this location during the committed wave to attempt a full commit operation before reporting results back to its initiator. **blank=** This LUW ended in a rollback request

Relative offset	Field	Format	Description
226	LUW disposition	Char (2)	The overall disposition of the LUW: **RO =** This location and all downstream locations voted read-only. These resources were not committed or rolled back because they were not changed during the LUW. It is not known whether the other locations in the sync-point tree committed or rolled back. **CM =** All resources committed. No errors have been detected to this point. If the Resync In Progress Indicator field is **N**, the LUW has completely committed. Otherwise, resynchronization is still going on to assure this location that other locations committed completely. **CF =** An attempt was made to commit all resources, but one or more errors have occurred. The job log, QHST, and QSYSOPR *MSGQ can be checked to determine the errors. **RB =** All resources rolled back successfully. **RF =** An attempt was made to roll back all resources, but one or more errors have occurred. The job log, QHST, and QSYSOPR *MSGQ can be checked to determine the errors. **HD =** Heuristic damage has occurred. This means one of two things: 1. Some of the resources at this location or downstream locations committed while others rolled back because an operator performed a heuristic commit operation or rollback operation. 2. An unexpected error occurred while committing or rolling back resources at this location or downstream locations due to a hardware or software problem. When heuristic damage occurs, the following LUW journal entry records can be checked to learn the status of the changes made during the LUW to individual resources: **LCL =** The Record I/O State field indicates the status of the record I/O performed on files journaled to the journal related to that location. **API =** The API State field indicates the status of that API Commitment Resource. **DDL =** The DDL State field indicates the status of that SQL Object Change. **RMT =** The Resource State field indicates the status of the resources at the remote location.

Relative offset	Field	Format	Description
228	Heuristic Operation Indicator	Char (1)	Whether a heuristic commit or rollback operation occurred at this location while a commit request was being performed for this LUW: **blank** No heuristic operation occurred. **C =** A heuristic commit operation occurred. **R =** A heuristic rollback operation occurred. A heuristic commit operation or rollback operation means that the operator took explicit action (while this location was waiting for the commit or rollback decision from the initiator or the last agent) to commit or to roll back the resources at this location and all prepared downstream locations. Heuristic operations can result in some resources committing while others roll back. The LUW Disposition field can be checked to see if this has happened (it would be **HD**). The Resync In Progress Indicator field can also be checked. If it is **O**, heuristic damage might have occurred or it might still occur because the state of the resources at the locations where resynchronization is still going on is unknown. Messages are written to the history log and to the system database server job logs when the resynchronization processes complete to indicate whether damage occurred. If damage occurs, messages are also sent to the system operator when it is detected.
229	Resync in progress indicator	Char (1)	Whether resync to one or more remote locations was still ongoing when the LUW ended: **N =** Either no resynchronization was required during this LUW, or it was required and completed before the LUW ended. **O =** Resynchronization was going on with one or more of the locations. This can occur only if the WAIT_FOR_OUTCOME synchronization point option is NO, or if the LUW was interrupted by job or system end.
230	Wait for outcome	Char (1)	The value of the Wait for outcome commitment option. This indicates whether to wait for resynchronization to complete if a communication or system failure occurs during a commit or rollback. **Y =** **Wait** for outcome. **L =** **Wait** for outcome during commits initiated by this commitment definition or during commits initiated at a system that does not support presumed abort. Inherit the initiator's wait for outcome value during commits initiated at a system that supports presumed abort **N =** Do **not** wait for outcome. **U =** Do **not** wait for outcome during commits initiated by this commitment definition or during commits initiated at a system that does not support presumed abort. Inherit the initiator's wait for outcome value during commits initiated at a system that supports presumed abort.

Relative offset	Field	Format	Description
231	Action if problems	Char (1)	The value of the Action if problems commitment option. This indicates whether to commit or rollback when problems occur during a two-phase commit. **R =** Rollback if problems occur. **C =** Commit if problems occur.
232	Vote read-only permitted	Char (1)	The value of the Vote read-only permitted commitment option. This indicates whether this commitment definition is allowed to return a read-only vote to a remote initiator during a two-phase commit. **N =** Do **not** allow a read-only vote. **Y =** **Allow** a read-only vote.
233	Action if ENDJOB	Char (1)	The value of the Action if ENDJOB commitment option. This indicates the action to take for changes associated with the LUW when the job the LUW is a part of is ended. **W =** Wait to allow normal processing of the LUW to complete. **R =** Rollback during ENDJOB. **C =** Commit during ENDJOB.
234	OK to leave out	Char (1)	The value of the OK to leave out commitment option. This indicates whether this location is allowed to be left out during the next commit/rollback if no activity occurred to this location during the LUW. **N =** Do **not** leave this location out of the next commit or rollback operation. **Y =** It is **OK** to leave this location out of the next commit or rollback operation.
235	Last agent permitted	Char (1)	The value of the Last agent permitted commitment option. This indicates whether last agent optimization may be used. **S =** The system **is** allowed to select a last agent. **N =** The system is **not** allowed to select a last agent.
236	Accept vote reliable	Char (1)	The value of the Accept vote reliable commitment option. This indicates whether the vote reliable indicator received from agents during a commit operation is accepted by this location. If an agent votes reliable, and this location accepts it, control is returned to the application before the committed wave is completed for that agent. If this location does not accept vote reliable, control is returned to the application only after the LUW is completely committed or rolled back. **Y =** **Accept** the vote reliable indicator from agents during commit operations. **N =** Do **not** accept the vote reliable indicator from agents during commit operations.

Relative offset	Field	Format	Description
237	Resolved wait for outcome value	Char (1)	This indicates the actual wait for outcome value that was used during the commit or rollback of this LUW. If the Wait for outcome commitment option is L or U, this value might have been inherited from this location's initiator. **Y =** Wait for outcome of resynchronization. **N =** Do not wait for outcome of resynchronization.
238	XA transaction manager	Char (10)	If this was an X/Open transaction, this is the name of the XA Transaction Manager that was specified on the db2xa_open API. This field will be hex zeros if this was not an XA transaction.
248	XID	Char (140)	If this was an X/Open Transaction, this is the X/Open Transaction Identifier associated with this transaction. This field will be hex zeros if this was not an X/Open transaction, or if it was an X/Open local transaction. The format of this field is as follows: Bin(31) format identifier Bin(31) global transaction identifier length Bin(31) branch qualifier length Char (128) XID value
388	Default journal commit cycle ID long	Zoned (20,0)	The commit cycle identifier for the default journal for this LUW. This is 0 if no commit cycle was started for this journal during this LUW.
408	Reserved	Char (9)	Reserved for future use.
Note: [1]The format for this field is in the description.			

Logical unit of work (C LW) journal entry - local record

Relative offset	Field	Format	Description
1	Record type	Char (4)	Type of record: **LCL =** Local location record.
5	Record length	Bin (15)	Length of record. Currently 48 for LCL record.
7	Record position	(4)[1]	This identifies the position in the LUW journal entry where this record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains this record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains this record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where this record starts within this journal entry. This is the number of bytes past the beginning of the entry where this record starts. For example, 0 means the first byte in the entry.

Relative offset	Field	Format	Description
11	Next local location position	(4)[1]	This identifies the position in the LUW journal entry where the next LCL record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 indicates that this is the last local location.
15	First resource position	(4)[1]	This identifies the position in the LUW journal entry where the first API or DDL record starts for this location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry.
19	Record I/O state	Char (2)	Indicates whether the record I/O performed during this LUW for files journaled to the journal related to this location was committed or rolled back successfully: **CS** Record I/O for this location was committed successfully. **RS =** Record I/O for this location was rolled back successfully **CF =** An attempt to commit record I/O for this location failed. **RF =** An attempt to rollback record I/O for this location failed. **blank=** This is the location with no journal so there is no record I/O associated with it.
21	Journal	Char (20)	Journal related to this location: • Char (10): Journal name (blank if this is the location with no journal) • Char (10): Journal library (blank if this is the location with no journal)

Relative offset	Field	Format	Description
41	Commit Cycle ID	Bin (31)	The commit cycle identifier for the journal. This is 0 for the location with no journal. It might be 0 for the location related to the default journal if there were no resources for that location during this LUW. This is -1 if the actual commit cycle identifier value is larger than 2 147 483 647. The Default Journal Commit Cycle ID Long field always contains the correct value.
45	Default journal flag	Char (1)	Indicates whether the journal related to this location is the default journal: **Y =** It is the default journal. **N =** It is not the default journal.
46	Commit Cycle ID Long	Zoned (20,0)	The commit cycle identifier for the journal. This is 0 for the location with no journal. It might be 0 for the location related to the default journal if there were no resources for that location during this LUW.
66	Reserved	Char (15)	Reserved for future use.
Note: [1]The format for this field is in the description.			

Logical unit of work (C LW) journal entry - RMT record

Relative offset	Field	Format	Description
1	Record Type	Char (4)	Remote Location (RMT) record.
5	Record Length	Bin (15)	RMT record is currently 128.
7	Record Position	(4)[1]	This identifies the position in the LUW journal entry where this record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains this record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains this record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where this record starts within this journal entry. This is the number of bytes past the beginning of the entry where this record starts. For example, 0 means the first byte in the entry.

Relative offset	Field	Format	Description
11	Next Remote Location Position	(4)[1]	This identifies the position in the LUW journal entry where the next RMT record starts. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 indicates that this is the last remote location.
15	First Resource Position	(4)[1]	This identifies the position in the LUW journal entry where the first DDM record starts for this location. It is made up of two numbers: • Bin (15): The relative number of the journal entry that contains the record. If the LUW journal entry is greater than 32K-1 bytes, multiple entries are actually sent to the journal. This number represents which of these actual journal entries contains the record (1 for the first, 2 for the second, and so on). Note that this is not the actual journal entry sequence number. • Bin (15): The offset where the record starts within this journal entry. This is the number of bytes past the beginning of the entry where the record starts. For example, 0 means the first byte in the entry. Position 0 0 indicates that there are no DDM records for this location.
19	Remote Location Information	Char (54)	Identification of the remote location and communication information for this location: • Char (10): Remote Location name • Char (10): Device name • Char (10): Mode • Char (8): Remote network ID • Char (8): Conversation correlator network ID • Char (8): Transaction program name
73	Relational Database Name	Char (18)	The name of the relational database opened at this remote location (blank if no relational database has been opened).
91	Conversation Deallocation Flag	Char (1)	Whether the conversation was deallocated because of this LUW: **N =** This conversation is still active. **Y =** This conversation was deallocated because the LUW committed, the system ended, a resource failed, or an unbind was performed.
92	Commit Protocol	Char (1)	The commit protocol for the resources at this location: **1 =** The resources are one-phase. **2 =** The resources are two-phase.

Relative offset	Field	Format	Description
93	Resource Usage	Char (2)	The currently allowed access for this resource. The allowed access for some resources can change from one LUW to another depending on whether one-phase resources are registered: **RO =** This resource is currently read-only. Updates were not made during the LUW. **UP =** This resource is currently able to be updated. Updates might or might not have been made during the LUW. **Note:** This does not indicate whether updates were actually made during the LUW. It indicates only whether updates are allowed, given the other resources currently registered.
95	Resource State	Char (2)	The state of the resources at this location: **CS =** The resources were committed successfully. **CF =** An attempt to commit the resources failed. This value is only used for one-phase locations. **RS =** The resources were rolled back successfully. **RF =** An attempt to rollback the resources failed. This value is only used for one-phase locations. **NC =** The resources had no changes for the current transaction. **FC =** A communications failure occurred for this location. It is not known whether resources at the location committed or rolled back. **HC =** The resources were heuristically committed. **HR =** The resources were heuristically rolled back. **HM =** Heuristic damage was detected at this location. Some of the resources at the location, or locations further downstream, committed while others rolled back. **ER =** An unexpected error occurred while communicating with this location. This is due to a hardware or software problem. The state of the resources is unknown. **RI =** We have not yet learned the state of the resources because resync is still ongoing.
97	Allocator Flag	Char (1)	Indicates whether this is the allocator location, for example, the location that called the transaction program running on this system: **Y =** This location is the allocator. **N =** This location is not the allocator.

Relative offset	Field	Format	Description
98	Remote Last Agent Flag	Char (1)	Indicates whether this location was selected as the last agent if a commit request was performed to end this LUW: **Y =** This is the last agent. **N =** This is not the last agent. **Note:** A last agent will not be selected at this location unless the Partner Role field in the HDR record is **I** or **L**.
99	Two-phase protocol	Char (1)	The two-phase commit protocol options supported at this location. **0 =** Two-phase commit protocols are not supported. **1 =** Two-phase commit presumed nothing protocols are supported. **2 =** Two-phase commit presumed abort protocols are supported.
100	Resync initiator	Char (1)	If resync with this location is still ongoing (the Resource State field is RI), this value indicates whether the local location is initiating the resync attempts. **I =** The local system is initiating resync with this remote location. **N =** Resync is not being performed with this remote location. **W =** The local system is waiting for resync to be initiated from this remote location.
101	Voted reliable	Char (1)	Whether this location voted reliable during the commit of this LUW. **Y =** The location voted reliable. **N =** The location did not vote reliable.
102	OK to leave out	Char (1)	Whether this location indicated it may be left out of the next commit or rollback operation if no communications flows occur to that location during the next LUW. **Y =** The location indicated it may be left out. **N =** The location indicated it may not be left out.
103	Left out	Char (1)	Whether this location was left out of the LUW that was just committed or rolled back. **Y =** The location was left out. **N =** The location was not left out.

Relative offset	Field	Format	Description
104	Initiator Flag	Char (1)	Indicates whether this location is the initiator location, i.e. the location that sent the commit or rollback request to this system. **Y =** The location is the initiator. **N =** The location is **not** the initiator. **Note:** The system cannot determine the initiator location if the initiator does not support two-phase commit protocols. This field will always be set to N for locations that do not support two-phase commit protocols.
105	Reserved	Char (24)	Reserved for future use.
Note: [1]The format for this field is in the description.			

Moving and renaming objects (D FM, D FN, E EM, E EN, F MM, F MN, F PM, F PN, Q QM, Q QN) journal entries

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Journal identifier (JOJID)	Char (10)	Records for the entries will have a journal identifier. The JID is not provided with the *TYPE1, *TYPE2, and *TYPE3 formats. It can be used with the QJORJIDI API.
Entry-specific data. This data appears as one field in the standard output formats:			
1	Object Name Before	Char (10)	The name of the object before the object was moved or renamed.
11	Library Name Before	Char (10)	The name of the library before the object was moved or renamed.
21	Member Name Before	Char (10)	The name of the member before it was moved or renamed. This field is blank if the object is not a physical database file.
31	Object Name After	Char (10)	The name of the object after the object was moved or renamed.
41	Library Name After	Char (10)	The name of the library after the object was moved or renamed.
51	Member Name After	Char (10)	The name of the member after it was moved or renamed. This field is blank if the object is not a physical database file.
61	Internal data	Char (*)	Internal system information.

Object level (D AC, D CG, D CT, D DC, D DT, D GC, D GO, D GT, D RV, D TC, D TD, D TG, D TQ, F DM, F MC) journal entries[1]

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Object name	Char (10)	The name of the object that was operated on.
11	Library Name	Char (10)	The name of the library for the object that was operated on.
21	Member Name	Char (10)	The name of the member that was operated on, if applicable. This field is blank if it does not apply.
31	Reserved	Char (78)	Reserved.
» 109	Change Field Type	Char (1)	The type of Change File operation: 0 = SQL ALTER TABLE 1 = CHGPF, CHGLF, or CHGSRCPF CL command 2 = Miscellaneous change file operations 3 = SQL DELETE FROM table (without a WHERE clause) The type of Change Trigger operation: 4 = Disable Trigger 5 = Enable Trigger 6 = Miscellaneous change trigger operations This field is not applicable if the entry type is not CG or TG. If the jounal entry is CT or MC, these subtype values are returned: 7 = Restore 8 = CPYF CRTFILE(*YES) or CRTDUPOBJ 9 = Other Create This field is not applicable if the entry type is not CG, CT, MC, or TG. «
110	Reserved	Char (3)	Reserved
» 113	Length of Trigger Library Name	Bin (15)	The length of the trigger library name for a Change Trigger operation. Contains 0 if the Change Trigger operation includes multiple triggers. This field is not applicable if the entry type is not TG.
115	Offset to Trigger Library Name	Bin (31)	The offset to the trigger library name for a Change Trigger operation from the beginning of the journal entry specific data. Contains hex zeros if the Change Trigger operation includes multiple triggers. This field is not applicable if the entry type is not TG.
119	Length of Trigger Name	Bin (15)	Length of the trigger name for a Change Trigger operation. Contains 0 if the Change Trigger operation includes multiple triggers. This field is not applicable if the entry type is not TG.
121	Offset to Trigger Name	Bin (31)	The offset to the trigger name for a Change Trigger operation from the beginning of the journal entry specific data. Contains hex zeros if the Change Trigger operation includes multiple triggers. This field is not applicable if the entry type is not TG.
125	Internal data	Char (*)	Internal system information. «

Notes:

1. This data does not apply to integrated file system objects.
2. If the data for these entries exceeds 32 KB, then a pointer is returned to the actual data when the entry is retrieved using an option to return pointers. If the return pointer option is not used, then *POINTER is returned for the entry-specific data.

» Object restored (B FR, D DZ, E EL, F MR, J RR, Q QZ) and receiver saved (J RS) journal entries «

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Journal identifier (JOJID)	Char (10)	Records for the entries will have a journal identifier. The JID is not provided with the *TYPE1, *TYPE2, and *TYPE3 formats. It can be used with the QJORJIDI API.
Entry-specific data. This data appears as one field in the standard output formats:			
1	Media type	Char (3)	The type of media used for the save or restore operation: **DKT=** Diskette **OPT=** Optical **SAV=** Save file **TAP=** Tape
4	First Volume ID	Char (6)	The ID of the first volume used. The optical volume ID might contain up to 32 characters of which the first six characters are displayed.
10	Start Save or Restore Date	Char (6)[1]	The date the save or restore operation was started. The date is in the format of the DATFMT attribute of the job that performed the save or restore operation.
16	Start Save or Restore Time	Zoned (6,0)	The time the save or restore operation was started.
22	Update History	Char (1)	Whether the save history is updated: **0 =** UPDHST(*NO) specified on save command. **1 =** UPDHST(*YES) specified on save command.
23	Save File Name	Char (10)	The name of the save file used for the operation. This field is blank if a save file was not used.
33	Save File Library	Char (10)	The name of the library for the save file. This field is blank if a save file was not used.
43	Media file identifier	Char (16)	File identifier for the integrated file system object on the media. This applies only to B FR entries.
59	Restored file identifier	Char (16)	File identifier for the restored integrated file system object. This applies only to B FR entries.
75	Restored over file identifier	Char (16)	File identifier for the integrated file system object that was restored over. This applies only to B FR entries.
Note: [1]See to the fixed-length portion of the journal entry for any information pertaining to the century of this date.			

Object saved (B FS, D DH, E ES, F MS, Q QY) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Media type	Char (3)	The type of media used to save the object: **DKT =** Diskette **OPT =** Optical **SAV =** Save file **TAP =** Tape
4	First Volume ID	Char (6)	The ID of the first volume used to save the object The optical volume ID might contain up to 32 characters of which the first six characters are displayed.
10	Start Save Date	Char (6)[1]	The date the save operation was started. The date is in the format of the DATFMT attribute of the job that saved the object.
16	Start Save Time	Zoned (6,0)	The time the save operation was started.
22	Update History	Char (1)	Whether the save history is updated: **0 =** UPDHST(*NO) specified on save command. **1 =** UPDHST(*YES) specified on save command.
23	Save File Name	Char (10)	The name of the save file used for the operation. This field is blank if a save file was not used.
33	Save File Library	Char (10)	The name of the library for the save file. This field is blank if a save file was not used.
43	Save Active Value	Char (10)	The value specified for the SAVACT parameter on the SAVOBJ, SAVCHGOBJ, SAV, or SAVLIB command.
53	Start Save Active Date	Char (6)[1]	For a save-while-active operation, this is the date when checkpoint processing was completed for the object. For a normal save operation, this is the same as the start date.
59	Start Save Active Time	Zoned (6,0)	For a save-while-active operation, this is the time when checkpoint processing was completed for the object. For a normal save operation, this is the same as the start time.
65	Primary Receiver Name	Char (10)	The name of the first of dual receivers that contains the start-of-save entry.
75	Primary Receiver Library	Char (10)	The name of the library containing the primary receiver.
85	Dual Receiver Name	Char (10)	The name of the second of dual receivers that contains the start-of-save entry. This entry is blank if only a single receiver was used when the start-of-save entry was added.
95	Dual Receiver Library	Char (10)	The name of the library containing the dual receiver. This entry is blank if only a single receiver was used when the start-of-save entry was added.
» 105	Sequence number of matching start-of-save entry	Zoned (10, 0)	For a save-while-active operation, the sequence number of the corresponding start-of-save entry. For a normal save operation, this is the sequence number of the current object saved entry. A -1 is returned if the sequence number is greater than 9 999 999 999. If -1, see Large sequence number of matching start-of-save entry. «
115	File ID of object or reserved	Char (16)	The file identifier for the object for B FS entries, otherwise blank.

Relative offset	Field	Format	Description
» 131	Large sequence number of matching start-of-save entry	Char (20)	For a save-while-active operation, the sequence number of the corresponding start-of-save entry. For a normal save operation, this is the sequence number of the current object saved entry.
151	Library ASP Device	Char (10)	The ASP device on which the library that contains the primary receiver resides. «
Notes: 1. See the fixed-length portion of the journal entry for any information pertaining to the century of this date. 2. If an object was saved using the save-while-active function, the saved copy of the object includes all of the changes found in the journal entries up to the corresponding object start of save-while-active entry. For more information see the layout for the Start of save-while-active journal entries. 3. If an object was NOT saved using the save-while-active function, the saved copy of the object includes all of the changes found in the journal entries up to the corresponding object saved entry. For more information see the layout for Object saved journal entries.			

Received data queue, has key (Q QL) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Reserved	Char (18)	Reserved for future use.
19	Key length	Bin (16)	The number of characters in the key.
21	Key order	Char (2)	The Key Order is as follows: **GT =** Greater than **LT =** Less than **NE =** Not equal **EQ =** Equal **GE =** Greater than or equal **LE =** Less than or equal
23	Key	Char (*)	The data to be used to receive a message from the data queue.

ROLLBACK (C RB) journal entry

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Job name (JOJOB)	Char (10)	Blank if the entry was added during an IPL vary on of an independent disk pool.
	Program name (JOPGM)	Char (10)	Blank if the entry was added during an IPL or vary on of an independent disk pool.

Relative offset	Field	Format	Description
	Flag (JOFLAG)	Char (1)	How the rollback operation was initiated and whether it was successful: **0 =** All record-level changes were rolled back for a rollback operation initiated by a user. **1 =** Not all record-level changes were successfully rolled back for a rollback operation initiated by a user. **2 =** All record-level changes were rolled back for a rollback operation initiated by the operating system. **3 =** Not all record-level changes were rolled back for a rollback operation initiated by the operating system.

» Rollback ended early (C CN, F C1) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	User profile	Char (10)	The user profile that requested to end the rollback.
11	Process	Char (26)	The process that requested to end the rollback. «

RGZPFM (F RG) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	File Name	Char (10)	The name of the file specified for the KEYFILE parameter on the RGZPFM command. If KEYFILE(*NONE) was specified, this field is blank.
11	Library Name	Char (10)	The name of the library specified in the KEYFILE parameter of the RGZPFM command. If KEYFILE(*NONE) was specified, this field is blank.
21	Member Name	Char (10)	The name of the member specified in the KEYFILE parameter of the RGZPFM command. If KEYFILE(*NONE) was specified, this field is blank.

Savepoint released (C SQ) and savepoint rolled back (C SU) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Sequence number	Char (20)	The sequence number where the savepoint was established

Send data queue, has key (Q QK) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
» 1	Data Length	Bin (32)	The length of the Data field (which is the last field in the ESD of this journal entry). When replicating a data queue entry with this journal entry, this length field should be specified on the QSNDDTAQ API in association with the Data field below. See the details in the description of the Data field.
5	Offset to Data	Bin (32)	Offset to the Data field (which is the last field in the ESD of this journal entry). The offset is calculated from the beginning of the entry-specific data (ESD). «
9	Reserved	Char (2)	Reserved for future use.
11	Key Length	Bin (16)	The number of characters in a key.
13	Reserved	Char (4)	Reserved for future use.
17	Key	Char (*)	A prefix added to an entry by its sender.
	Reserved	Char (*)	Padding to align fields.
» Offset to data	Data	Char (*)	The first 16 bytes of the Data field are API information required by the Send Data Queue (QSNDDTAQ) API. When replicating a data queue entry with this journal entry, this entire Data field (including the 16 bytes of API information) must be passed to the QSNDDTAQ API when it is called with parameter eight (Data is from a journal entry) set to *YES. These 16 bytes are not placed on the data queue. The remainder of the Data field is placed on the data queue. «

Send data queue, no key (Q QS) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Reserved	Char (28)	Reserved for future use.
» 29	Data length	Bin (32)	The length of the Data field (which is the last field in the ESD of this journal entry). When replicating a data queue entry with this journal entry, this length field should be specified on the QSNDDTAQ API in association with the Data field below. See the details in the description of the Data field.
33	Data	Char (*)	The first 16 bytes of the Data field are API information required by the Send Data Queue (QSNDDTAQ) API. When replicating a data queue entry with this journal entry, this entire Data field (including the 16 bytes of API information) must be passed to the QSNDDTAQ API when it is called with parameter eight (Data is from a journal entry) set to *YES. These 16 bytes are not placed on the data queue. The remainder of the Data field is placed on the data queue. «

Start of save-while-active (B FW, D DW, E EW, F SS, Q QX) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			

Relative offset	Field	Format	Description
1	Media type	Char (3)	The type of media used to save the object: **DKT =** Diskette **OPT =** Optical **SAV =** Save file **TAP =** Tape
4	First Volume ID	Char (6)	The ID of the first volume used to save the object. The optical volume ID might contain up to 32 characters of which the first six characters are displayed.
10	Start Save Date	Char (6)[1]	The date the save operation was started. The date is in the format of the DATFMT attribute of the job that saved the object.
16	Start Save Time	Zoned (6,0)	The time the save operation was started.
22	Update History	Char (1)	Whether the save history is updated: **0 =** UPDHST(*NO) specified on the save command. **1 =** UPDHST(*YES) specified on the save command.
23	Save File Name	Char (10)	The name of the save file used for the operation. This field is blank if a save file was not used.
33	Save File Library	Char (10)	The name of the library for the save file. This field is blank if a save file was not used.
43	Save Active Value	Char (10)	The value specified for the SAVACT parameter on the SAVOBJ, SAVCHGOBJ, SAV, or SAVLIB command.
53	Save Active Date	Char (6)[1]	For a save-while-active operation, this is the date when checkpoint processing was completed for the object. For a normal save operation, this is the same as the start date.
59	Save Active Time	Char (6)	For a save-while-active operation, this is the time when checkpoint processing was completed for the object. For a normal save operation, this is the same as the start time.
65	Object File ID	Char (16)	The file identifier of the integrated file system object. This applies only to B FW entries.

Notes:

1. See the fixed-length portion of the journal entry for any information pertaining to the century of this date.
2. If an object was saved using the save-while-active function, the saved copy of the object includes all of the changes found in the journal entries up to the corresponding object start of save-while-active entry. For more information see the layout for Start of save-while-active journal entries.
3. If an object was NOT saved using the save-while-active function, the saved copy of the object includes all of the changes found in the journal entries up to the corresponding object saved entry. See the entry-specific data for Object saved journal entries.

Start journal (B JT, D JF, E EG, F JM, Q QB) journal entries

Relative offset	Field	Format	Description
Specific values for this entry type:			
	Flag (JOFLAG)	Char (1)	Indicates the type of images selected: **0 =** After images are journaled. **1 =** Before and after images are journaled.
Entry-specific data. This data appears as one field in the standard output formats:			
1	Omit journal entry	Char (1)	Indicates the value of the OMTJRNE parameter on the Start Journal command. **0 =** No entries are omitted from journaling. **1 =** Open and Close (*FILE), or Open, Close, and Force (*DIR or *STMF) entries are not journaled.
2	New object inherit journaling	Char (1)	Specifies whether journaling starts automatically for new objects created in the directory. **0 =** No or does not apply **1 =** Yes
3	Reserved	Char (6)	Reserved field
9	File identifier	Char (16)	The file identifier for the integrated file system object. This only applies to B JT entries.
25	Path name	Char (*)	The path name information optionally follows the file identifier. This only applies to B JT entries. For path name information, see Path name.

Usage limit changed (L LL) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Product ID	Char (7)	The ID of the product whose usage limit was changed.
8	License Term	Char (6)	The term of the license.
14	Feature	Char (4)	The product feature code.
18	Previous Usage Limit	Zoned (6,0)	The usage limit before the change.
24	Current Usage Limit	Zoned (6,0)	The usage limit after the change.
30	Old Expiration Date.	Char (7)	The expiration date before the change.
37	New Expiration Date.	Char (7)	The expiration date after the change.

Usage limit exceeded (L LU) journal entry

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Product ID	Char (7)	The ID of the product whose usage limit was exceeded.
8	License Term	Char (6)	The term of the license.
14	Feature	Char (4)	The product feature code.

Relative offset	Field	Format	Description
18	Usage Limit	Zoned (6,0)	The usage limit for the product.
24	Request Flag	Char (1)	Whether the request was successful: **0 =** License request was successful. **1 =** License request was not successful.
25	Number of Licensed Users	Zoned (6,0)	The number of users currently licensed for the product.
31	Licensed User Name	Char (26) x 100	The names of up to 100 users who are licensed for the product.

Update data area (E EA, E EB) journal entries

Relative offset	Field	Format	Description
Entry-specific data. This data appears as one field in the standard output formats:			
1	Starting position	Bin (32)	Starting position of change as specified by the user (1 for decimal).
5	Length of change	Bin (32)	Length of change to be applied as specified by the user.
9	Number	Bin (32)	Number of decimal positions as specified by the user.
13	Offset to change	Bin (32)	Offset to change value field from the beginning of the entry-specific data (ESD).
17	Type	Char (10)	Type of data area. Data area types are *CHAR, *DEC, and *LGL.
	Padding for alignment	Char (*)	Padding to align fields.
Offset to change	Change value	Char (*)	Value of the change.

Common fields

The following tables contain fields of entry-specific data that are common to more than one journal entry layout.

Journal information

This table contains entry-specific data for journal information. For an explanation of these fields, see the Get Attributes (Qp0lGetAttr()) API.

Relative offset	Field	Format	Description
1	Journaling status	Char (1)	Indicates whether the object is journaled
2	Options	Char (1)	The journaling options or attributes
3	JID	Char (10)	The journal identifier
13	Journal name	Char (10)	The journal name
23	Journal library	Char (10)	The journal library
33	Time journaling was last started	Bin (32)	Time journaling was last started

Object name

This table contains entry-specific data for the name of an integrated file system object. For more information about the object name see Path name format.

Relative offset	Field	Format	Description
1	Length	Bin (32)	The length of the object name field.
5	Path name CCSID	Bin (32)	The coded character set identifier (CCSID) for the object name.
9	Object name country ID	Char (2)	The country identifier for national language support.
11	Object name language ID	Char (3)	The language identifier for national language support.
14	Reserved.	Char (3)	Reserved. This field contains all hex zeros.
17	Object name	Char (*)	The object name. The field is of variable length.

Path name

This table contains entry-specific data for the path name of an integrated file system object. For more information about path name see Path name format.

Relative offset	Field	Format	Description
1	Path indicator	Char (1)	The absolute or relative path indicator. This field uses one of the following values: **0 =** The path contains an absolute path name. The Relative directory FID field is hex zeros. **1 =** The path contains a relative path name. The Relative directory FID field is valid and can be used to form a complete path name.
2	Relative directory file ID	Char (16)	The file identifier for the directory that contains the object indicated in the path name filed.
18	Path name CCSID	Bin (31)	The coded character set identifier (CCSID) for the path name.
22	Path name country ID	Char (2)	The country identifier for national language support.
24	Path name language ID	Char (3)	The language identifier for national language support.
27	Reserved.	Char (3)	Reserved. This field contains all hex zeros.
30	Path name type	Bin (32)	The path name type uses one of the following values: **0 =** The path name is a character string with a one byte delimiter. **2 =** The path name is a character string with a two byte delimiter.
34	Path length	Bin (31)	The path length.
38	Path name delimiter	Char (2)	The path name delimiter.
40	Reserved	Char (10)	Reserved. Set to blanks.
50	Path name	Char (*)	The object path name. This field is of variable length.

Symbolic link contents

Relative offset	Field	Format	Description
1	Contents included	Char (1)	Indicates if the entire symbolic link contents are included in the Contents field. This field uses of the following values: **0 =** The entire symbolic link contents cannot be included in the Contents field. The symbolic link contents are truncated in this entry. **1 =** The entire symbolic link contents are included in the Contents field.
2	Contents CCSID	Bin (31)	The coded character set identifier (CCSID) for the symbolic link contents.
6	Contents country ID	Char (2)	The country identifier for national language support for the symbolic link contents.
8	Contents language ID	Char (3)	The language identifier for national language support for the symbolic link contents.
11	Reserved.	Char (3)	Reserved. This field contains all hex zeros.
14	Contents path type	Bin (32)	The contents path type uses one of the following values for the symbolic link contents: **0 =** The path name is a character string with a one byte delimiter. **2 =** The path name is a character string with a two byte delimiter.
18	Contents path length	Bin (31)	The path length for the symbolic link contents.
22	Contents path name delimiter	Char (2)	The path name delimiter for the symbolic link contents.
24	Reserved	Char (10)	Reserved. This field contains all hex zeros.
34	Symbolic link contents	Char (*)	The symbolic link contents. This field is of variable length.

Work with journal entry information

Every journal entry is stored internally in a compressed format and must be converted by the operating system to an external form before it can be shown to the user. You cannot change or access the journal entries directly. Not even the security officer can remove or change journal entries in a journal receiver.

You can use these journal entries to help you recover your objects or analyze changes that were made to the objects.

Following are the various ways that you can retrieve, display, and print journal entry information:

- Display and print journal entries
- Receive journal entries in an exit program
- Retrieve journal entries in a program
- Work with pointers in journal entries »
- Replay a database operation from a single journal entry «
- Considerations for entries containing minimized entry-specific data

Note: Read the Code example disclaimer for important legal information.

Display and print journal entries: Use the Display Journal (DSPJRN) command to display journal entries. The entries are displayed at a work station, printed, or written to an output file. You cannot directly access the journal entries in the form in which they are contained in the journal receivers.

The Journal entry information finder describes each type of journal entry and the information that it contains. It also provides links for topics that provide the layouts for the fixed-length portion and the variable-length portion of the journal entry. See the Display Journal (DSPJRN) Command Description for complete layouts for the model database output files that are provided by the system.

Often, to prepare for a recovery, you display or print the journal entries first. Journal code descriptions provides a description of each code. Use this list to help you analyze the journal entries and to do the following:

- Prepare for the recovery of a particular object. The list contains the information you need to specify the starting and ending points for applying and removing journaled changes.
- Determine the functions that have been performed on the objects that are being journaled (such as save and restore, clear, reorganize).
- Determine the functions that have been performed on the journal (such as attaching new journal receivers).
- Determine the functions that have been performed on the associated journal receivers (such as save and restore).
- Review the activity that has occurred on an object.
- Analyze journal entries for debugging or problem analysis.
- Analyze journal entries for an audit trail.

The DSPJRN command either can selectively list journal entries for a particular member of a file or list the entries for all files within a particular library. You can further identify journal entries by specifying other selection criteria such as:

- Journal entries for specific entry types or journal codes, such as U (user-created entries)
- Journal entries for a particular job, program, or file
- Commit cycle identifier
- Date and time
- Dependent entries (referential integrity, triggers, and entries that will be ignored during an Apply Journaled Changes (APYJRNCHG) or Remove Journaled Changes (RMVJRNCHG) operation)
- Any combination of these:

The online help describes all the parameters for the DSPJRN command. To view the help, type DSPJRN on a command line and press F1.

Specify journal codes

You can display entries that have specific journal codes, such as all file-member-level entries (F), all record-level entries (R), or all security entries (T). You specify journal codes in paired values. The first value in the pair is the journal code. The second value indicates whether the file selections you have specified apply when deciding to display entries with the journal code.

Following is an example:

```
DSPJRN JRN($JRNLIB/JRNA) FILE(CUSTLIB/FILEA)
       JRNCDE((F *ALLSLT) (R *ALLSLT)
             (U *IGNFLSLT))...
```

In this example, entries for the FILEA file with journal codes F and R are displayed if the entries meet all other selection criteria, such as date and time. Entries with journal code U are displayed regardless of whether they are for file FILEA, because ignore file selection (*IGNFLSLT) is specified for journal code U. Entries with journal code U must meet all other selection criteria, such as date and time, to be displayed.

Specify output

The following topics provide information about specifying output for journal entries:

- Output for journal entries directed to a workstation
- Output for journal entries directed to a database output file
- Format of database output files

Note: Read the Code example disclaimer for important legal information.

Output for journal entries directed to a workstation: If you direct the output from the Display Journal (DSPJRN) command to the requesting workstation, basic information about the journal entries appears. Use the roll key to display the next sequential set of entries.

» If you specify a receiver range that includes an attached journal receiver, and you specify TOENT(*LAST) or TOENTLRG(*LAST), the display shows last journal entries in the journal. Press the Page Down key to see any new journal entries that are added to the attached receiver since the last time you pressed the Page Down key. «

The attached journal receiver in receiver range refers to the journal receiver that was currently attached when the DSPJRN command was first issued. That journal receiver could be detached while you are looking at the data online. If that occurs, paging down does not display any entries added after that receiver was detached.

Output for journal entries directed to a database output file: If you direct the output from the Display Journal (DSPJRN) command to a database output file, you can further restrict the journal entries you want to process by creating logical files over the database output file.

Each journal entry occupies one record in the output file. Each has a fixed-length portion for standard files. Before-images and after-images occupy separate records. The ENTDTALEN parameter controls the length of the field that is used to contain the record image. The ENTDTALEN parameter also controls whether the field is a fixed or variable length field. If the journal entry is smaller than the output file record, the journal entry is padded with blanks. If the journal entry is larger than the output file record, the remainder of the journal entry is truncated, and the system issues a warning message. To avoid truncation, specify the maximum record length in your files for the ENTDTALEN parameter on the DSPJRN command or specify *CALC for the ENTDTALEN parameter to allow the system to calculate the length of the specific data field so no entry is truncated.

If you write journal entries to a database output file, you can write application programs that will process the data to:

- Write your own apply program.
- Correct data that has been incorrectly updated.
- Remove or review all changes that were made by a particular program.

If you remove all changes that were made by a particular program, you could remove some valid updates. For example, assume that two work station users are using the same program to update an object, and one user enters some data that is not valid. If you remove all invalid data changes that are made by that program, you also remove the valid data that is entered by the other work station user.

Format of database output files: When you direct the output of the Display Journal (DSPJRN) command to a database file, the system creates the output file records in a standard format. The system creates the database file in one of these standard formats that are determined by the value that is specified for the OUTFILFMT parameter:

- *TYPE1
- *TYPE2
- *TYPE3
- *TYPE4
- *TYPE5

Fixed-length portion of the journal entry Has a complete description of these formats.

You can create an output file to hold the output from the DSPJRN command, but the format has to match the format of one of the IBM[(R)]-supplied output files.

Processing journal entry data

There are many ways to work with the journal entry data, including the entry-specific data, depending on the command that you use to process the journal entry data.

- Use your high-level language (HLL) to subdivide the fields into subfields.
- Use the Retrieve Journal Entry (RTVJRNE) command and the substring built-in function.
- Use the Receive Journal Entry (RCVJRNE) command and the substring built-in function.
- Use the Retrieve Journal Entries (QjoRetrieveJournalEntries) API and map out the data that is returned.

Analyzing your journal activity

You can use the DSPJRN command to help analyze your journal entries. For example, you could determine how many of each type of entry (such as add or update) was done for a specific file or by a specific user.

Receive journal entries in an exit program: You can write a program to receive journal entries as they are written to the journal receiver. When you use the Receive Journal Entry (RCVJRNE) command, you can specify a user-defined program, called an **exit program**, to receive journal entries. The program can, for example, write the entries to tape or to an OS/400[(R)] intersystem communications function (ICF) file that sends them to a backup system. You can use the received entries to update a backup copy of the primary object on the backup system. You cannot use these received entries with system-supplied recovery commands (Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG)) to update your objects because the RCVJRNE command converts the entries to their external form. You must write your own program to apply the changes that are contained in the entries to the objects.

The RCVJRNE command supports the same selection criteria as the Display Journal (DSPJRN) command. You can specify which entries go to the exit program.

For example, you can choose not to receive journal entries that are generated by the action of trigger programs or referential constraints. If you have a user-written program that updates the files on a second system with the journal entries, you probably want to specify DEPENT(*NONE). The actions performed by trigger programs or referential constraints are duplicated automatically on the second system if your database definitions are the same and you replay the original file operations.

You can specify DELAY(*NEXTENT) to have journal entries sent to your program as soon as they are written to the journal receiver. You can also specify a time interval. When that interval ends, the exit program is called. Either new entries are sent or an indicator is sent that there are no new entries.

- Exit program to receive journal entries
  Use the parameters in this topic to determine how the exit program will receive journal entries.
- Request block mode
  Use block mode to specify whether the system will be sending one or more journal entries to the exit program and specifies the block length of the buffer passed to the exit program.

Exit program to receive journal entries: You use two parameters to communicate between your exit program and the system when you are receiving journal entries. The system uses the first parameter for the contents of one or more journal entries that it is passing to the exit program. The exit program uses the first parameter to indicate the block length if the exit program requests block mode.

The system and the exit program use the second parameter to communicate about status changes, such as requesting block mode or ending the RCVJRNE command. The second parameter is a character field that is three bytes long. Following are the possible values for the first byte of the second parameter:

Possible values for the first byte of the second parameter	
0	This value is passed from the system to the exit program. It indicates that no journal entry is being passed on this call of the exit program.
1	This value is passed from the system to the exit program. It indicates that a single journal entry is being passed on this call of the exit program. If the specified entry format is not *TYPEPTR or *JRNENTFMT, then the figure, First parameter of RCVJRNE command: Single-entry mode shows the layout of the first parameter. Otherwise, the layout is the same as returned to the Retrieve Journal Entries (QjoRetrieveJournalEntries) API interface.
2	This value is passed from the system to the exit program. It indicates that block mode is in effect. One or more journal entries are being passed on this call of the exit program. If the specified entry format is not *TYPEPTR or *JRNENTFMT, then the figure, First parameter of RCVJRNE command: Block mode shows the layout of the first parameter. Otherwise, the layout is the same as returned to the QjoRetrieveJournalEntries API interface.
3	This value is passed from the system to the exit program. It indicates that no journal entry is being passed on this call of the exit program because the journal receiver that was attached when the Receive Journal Entry (RCVJRNE) command was started is no longer attached. The system ends the RCVJRNE command after returning this value to the exit program.
4	No journal entry is passed on this call to the exit program, and no more entries can be passed unless the local or remote journal is activated. This value can only be passed to the exit program when receiving journal entries from the attached receiver of a local or remote journal. The journal state for the journal must be *INACTIVE.
8	This value is passed from the exit program to the system. It indicates that the system must begin block mode and pass multiple entries to the exit program. You can also specify block mode by using the BLKLEN parameter of the RCVJRNE command. If you specify a BLKLEN value other than *NONE, then specifying 8 in the first byte of the second parameter will have no impact and the first 5 bytes of the first parameter bill be ignored. However even if BLKLEN(*NONE) is specified, the system will begin block mode if you specify 8 for the first byte of the second parameter. See Request block mode for more information.
9	This value is passed from the exit program to the system. It indicates that the RCVJRNE command will be ended.

Possible Values for the Second Byte of the Second Parameter:	
N	This value is passed from the system to the exit program. Additional journal entries are not currently available to be passed after this call of the exit program, or the RCVJRNE command will end after this call of the exit program.
Y	This value is passed from the system to the exit program. Additional journal entries are currently available to be passed after this call of the exit program.

Possible values for the third byte of the second parameter:	
'00' x	One or more journal entries are being passed to the exit program and the object names in the fixed-length portion of each journal entry do not necessarily reflect the name of the object at the time the journal entry was deposited into the journal. This value is only returned when receiving journal entries from a journal receiver that was attached to a journal prior to V4R2M0.
0	No journal entries are currently being passed, so the information that is normally returned in this byte is not applicable.
1	One or more journal entries are being passed to the exit program. The object names in the fixed-length portion of each journal entry reflect the name of the object at the time the journal entry was deposited into the journal.
2	One or more journal entries are being passed to the exit program. The object names in the fixed-length portion of each journal entry do not necessarily reflect the name of the object at the time the journal entry was deposited into the journal. The object name in the fixed-length portion of the journal entry may be returned as a known name for the object prior to the journal entry being deposited into the journal. The object name in the fixed-length portion of the journal entry may also be returned as *UNKNOWN. This value will only be returned when receiving 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 Journal State (QjoChangeJournalState) API or Change Remote Journal (CHGRMTJRN) command is invoked and is currently replicating journal entries to the remote journal. After the call to the QjoChangeJournalState API or CHGRMTJRN command returns, the remote journal is maintained with a synchronous or asynchronous delivery mode, and the remote journal is no longer being caught up. Refer to Retrieve journal entries from a remote journal during the catch-up phase for more information.

Any information that is passed from the exit program to the system in the second byte or third byte is ignored.

The second byte of the second exit program parameter is provided whether journal entries are being processed as a single journal entry per call of the exit program, or as a block of journal entries per call.

When an N is passed to the exit program in the second byte of the second parameter indicated that no additional journal entries are currently available, it does not necessarily mean that when the exit program returns, that the RCVJRNE command will have to wait for additional journal entries to be deposited into the journal. By the time the exit program returns, additional journal entries may already be available and depending upon what was specified on the DELAY parameter, may or may not be immediately passed to the exit program. If DELAY(N) was specified the system will wait N seconds before passing the journal entries to the exit program. If DELAY(*NEXTENT) was specified, the journal entries will immediately be passed to the exit program.

Request block mode: When you request block mode, the system sends more than one journal entry to the exit program at a time. You can request block mode at any time. There are two ways that you can request block mode:

- Specify the BLKLEN parameter on the Receive Journal Entry (RCVJRNE) command
- Specify 8 for the value of the first byte of the second parameter of the exit program

BLKLEN parameter of the RCVJRNE command

When you specify the BLKLEN parameter of the RCVJRNE command you can select one of three values:

***NONE**
At most one journal entry will be sent to the exit program.

***CALC**
One or more journal entries will be passed to the exit program in a block. The length of the block passed (the first parameter passed to the exit program) is determined by the system and will be optimal.

block-length
Specify the length in kilobytes of the buffer passed to the exit program (EXITPGM parameter). Valid values range from 32 to 4000

If you specify BLKLEN(*CALC) or BLKLEN(block-length), specifying 8 in the first byte of the second parameter will have no impact and the first 5 bytes of the first parameter will be ignored.

Specify 8 for the value of the first byte of the second parameter of the exit program

When you specify 8 for the value of the first byte of the second parameter, you must specify the block length in the first 5 bytes of the first parameter as a zoned decimal (Zoned (5,0)) field. 99999 bytes is the maximum block size. After you have requested block mode, the system remains in block mode until the RCVJRNE processing is ended.

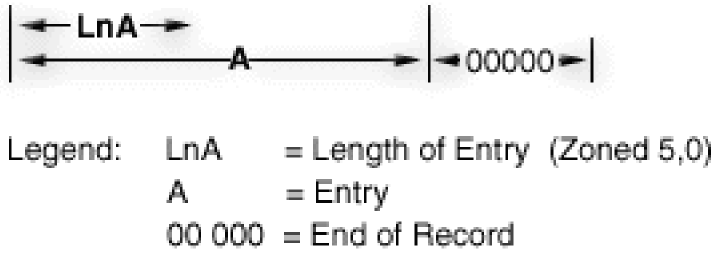
If you request block mode and the system is already using block mode, your request is ignored. You cannot change the size of the block from the size you specified when you first requested block mode.

Even if BLKLEN(*NONE) is specified, if you specify 8 for the value of the first byte of the second parameter, the system will use block mode.

Format of the first Parameter

If the specified entry format is not *TYPEPTR or *JRNENTFMT, and if you are using single-entry mode, the format of the first parameter looks like the following figure:

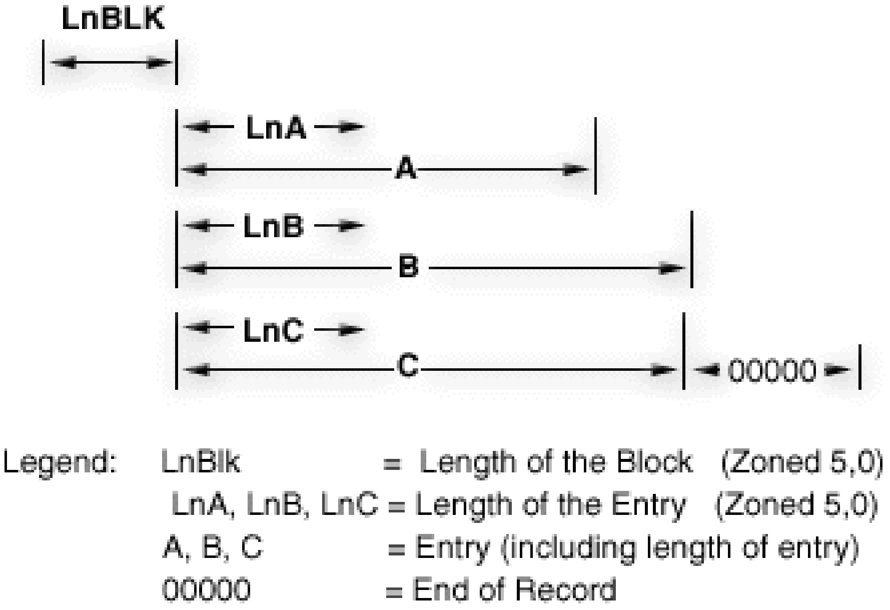
First parameter of RCVJRNE command: Single-entry mode



The first 5 bytes contains the length of the entry. The last 5 bytes contains all zeroes. The length of the entry does not include the 5 bytes of zeroes at the end of the record.

If the specified entry format is not *TYPEPTR or *JRNENTFMT, and if you are using block mode, the format of the first parameter looks like the following figure:

First parameter of RCVJRNE command: Block mode


LnBLK
LnA
A
LnB
B
LnC
C
00000
Legend: LnBlk = Length of the Block (Zoned 5,0)
LnA, LnB, LnC = Length of the Entry (Zoned 5,0)
A, B, C = Entry (including length of entry)
00000 = End of Record

The first 5 bytes contains the total length of the block. This length includes the 5 bytes for the total block length, the 5 bytes of the End of Record field at the end of the block, and all of the length and data fields in between. If no entry is being passed, this Block Length field contains zeroes. The block always ends with a 5-byte End of Record field containing zeroes.

If you specify BLKLEN(*NONE), then the system fills the block with as many complete entries as it can fit within the block size that you specified. The system does not send a partial entry to fill the block size. If the specified entry format is not *TYPEPTR or *JRNENTFMT, the maximum number of bytes that are available for the journal entries is 99989 bytes. 10 bytes in each block are reserved for the Block Length field and for the End of Record field. If the specified entry format is *TYPEPTR or *JRNENTFMT, the maximum number of bytes that are available is 99999 bytes.

If you specify a block size that is not valid, the system begins block mode but it sends only one journal entry per block. The system sends message CPD7095 to indicate that you have specified a block size that is not valid. If you specify a block size that is not valid or too small for a single journal entry, the system still returns at least one journal entry to the exit program. If the specified entry format is *TYPEPTR or *JRNENTFMT, the block size must be at least 13 bytes to be considered valid.

When the System Sends a Record

When block mode is in effect, the system uses the following rules to determine when to call the exit program:

- If the block does not contain any entries but the next entry would exceed the maximum size for the block, then the entry is placed into the block. The exit program is called. The system always passes at least one complete journal entry to the exit program.
- If the next entry to be put into the block would exceed the maximum size for the block and the current block has entries in it, then the current block of entries is passed to the exit program.
- If the current block has one or more entries in it and no additional entries in the journal meet the selection criteria, the current block of entries is passed to the exit program.

When in block mode, the specification for the DELAY parameter is used only when the current block is empty and no entries are currently available to be returned to the exit program.

Use ENTFMT(*TYPEPTR) or ENTFMT(*JRNENTFMT) with the RCVJRNE command

If the specified entry format is *TYPEPTR or *JRNENTFMT, the layout of the journal entry data is the same as the layout that is described in the QjoRetrieveJournalEntries API interface. The layout is the same for both single entry and block entry mode when you specify *TYPEPTR or *JRNENTFMT.

If you specify *TYPEPTR, the format will be the same as the RJNE0100 format of the QjoRetrieveJournalEntries API.

When you specify *TYPEPTR or *JRNENTFMT, the journal entry data may have pointers that will point to additional entry-specific data. See Work with pointers in journal entries for more information.

Retrieve journal entries in a program: You can use the Retrieve Journal Entry (RTVJRNE) command or the Retrieve Journal Entries (QjoRetrieveJournalEntries) API in a program to retrieve a journal entry and place it in a variable in the program.

You can also use the QjoRetrieveJournalEntries API to retrieve a journal entry and return data which can include pointers.

RTVJRNE command

Use the RTVJRNE command in a program to retrieve a journal entry and place it in variables in the program. You can retrieve the following:

- Sequence number
- Journal code
- Entry type
- Journal receiver name
- Library name for the journal receiver
- Journal entry-specific data

You can use this method to create programs to automate recovery. For layout of the fixed-length portion and variable-length portion of a journal entry see the Journal entry information finder.

For the format of the record for the RTVJRNE command, see the Retrieve Journal Entry (RTVJRNE)Command Description.

The QjoRetrieveJournalEntries API

The QjoRetrieveJournalEntries API allows you to retrieve journal entries into a receiver variable. The available journal entry information is similar to what is provided by using the Display Journal (DSPJRN), Receive Journal Entry (RCVJRNE), and Retrieve Journal Entry (RTVJRNE) commands. This API also provides additional journal entry data that cannot be retrieved with these commands. This additional data is accessed using pointers. Refer to Working with pointers in journal entries for more information.

Work with pointers in journal entries: Under certain conditions, not all of the journal entry data will be immediately retrievable from a journal entry. Instead, part of the journal entry information will include pointers to additional journal entry-specific data. These pointers will only be retrieved if you use following:

- Retrieve Journal Entries (QjoRetrieveJournalEntries) API
- The *TYPEPTR format on the Receive Journal Entry (RCVJRNE) command
- The *JRNENTFMT format on the RCVJRNE command (you must also specify the RTNPTR parameter for the RCVJRNE command)

In all other retrievals of journal entry data, *POINTER would be in the field where a pointer could exist. An incomplete data indicator has been added to indicate if the journal entry-specific data has data missing which can only be retrieved through a pointer

If the QjoRetrieveJournalEntries API or the *TYPEPTR or *JRNENTFMT format on RCVJRNE command is used and the incomplete data indicator field is 1, the journal entry-specific data will contain pointers. For all other interfaces, if the incomplete data indicator is 1, the journal entry-specific data will have the character string *POINTER in the field where an actual pointer would be placed if the API or *TYPEPTR or *JRNENTFMT interfaces were used. The incomplete data indicator field could be set to 1 if the journal entry-specific data exceeds 32766 bytes, or if the journal entry is associated with a database file which has one or more fields of data type BLOB (binary large object), CLOB (character large object), or DBCLOB (double-byte character large object). Use the Journal entry information finder to find which journal entry types can set the incomplete data indicator on.

These pointers can only be used with the V4R4M0 and later versions of the following languages:

- ILE/COBOL
- ILE/RPG
- ILE/C if the TERASPACE parameter is used when compiling the program. See WebSphere(R) Development Studio ILE C/C++ Programmer's Guide for information about using the TERASPACE parameter.

There are some considerations you need to be aware of when using the pointer data:

- The pointer can only be used by the process or job which retrieved or received the journal entry which contained the pointer. The pointer cannot be passed on to another job, nor can it be stored to use at a later date by a different job or process.
- The pointer will only give you read access to the additional data. Write operations to that pointer are not allowed.
- The data that is being pointed to actually resides in the journal receiver. Therefore, ensure that you protect the journal receiver from deletion until you use the data. To prevent a journal receiver from being deleted before the data is used, you can register an exit point for the Delete Journal Receiver (DLTJRNRCV) command. For more information, refer to Delete journal receivers.
- For files with fields of data type BLOB (binary large object), CLOB (character), or DBCLOB (double-byte character large object), use SQL to update the files.

If any journal entries are returned with pointers, the journal entry will also contain a pointer handle. This pointer handle must be used to free up any allocations associated with the pointer data once the pointer data has been used. The considerations for this pointer handle are as follows:

- Using the pointer data means any of the following:
  - Addressing the information and copying the addressed data to another object
  - Using the journal entry-specific data directly to modify another object. For example, using the data to update a database file with the journal entry which represents a database record update for a file which included LOBs.
  - Ignoring the additional data that is pointed to
- If you used the QjoRetrieveJournalEntries API, use the Delete Pointer Handle (QjoDeletePointerHandle) API to delete the pointer handle when you are done using it.
- If you use the RCVJRNE command with the RTNPTR(*SYSMNG) parameter, you must use the associated pointer prior to returning control from the exit program. The system will delete all pointer handles after the return from the exit program call.
- If you use the RCVJRNE command with the RTNPTR(*USRMNG) parameter, then it is your responsibility to use the Delete Pointer Handle (QjoDeletePointerHandle) API to delete the pointer handle when you are done using it.

Replay a database operation from a single journal entry: » Use the Replay Database Operation (QDBRPLAY) API to replay a database operation from a single journal entry. You can only use the QDBRPLAY API to replay journal entries for database physical-file objects. Also, the API does not run under commitment control even if the original journal entry was performed as part of a committable transaction.

Since these database journal entries can be quite large, use the Retrieve Journal Entries (QjoRetrieveJournalEntries) API to retrieve the journal entries.

You can use the QDBRPLAY API to replay the following journal entries. You can get more information about these journal entries in the Journal entry information finder.

Journal code	Entry type	Description
D	AC	Add Constraint
D	CG	Change File
D	CT	Create File
D	DC	Remove Constraint
D	DT	Delete File
D	FM	Move File
D	FN	Rename File
D	GC	Change Constraint
D	GO	Change Owner
D	GT	Grant File
D	RV	Revoke File
D	TC	Add Trigger
D	TD	Remove Trigger
D	TG	Change Trigger
D	TQ	Refresh Table
F	CB	Change Member
F	DM	Remove Member
F	MC	Add Member
F	MN	Rename Member
F	RM	Reorganize Member

Rename exit program

The QDBRPLAY API has an exit program that can change the names of the objects that are referenced in the journal entry. If 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 might be short names or long SQL names. The same name might 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 OS/400(R) and SQL rules for object names.

See Replay Database Operation (QDBRPLAY) API description for required parameters, authorities, and restrictions. See the following topics for more information about the QjoRetrieveJournalEntries API:
- Retrieve journal entries in a program
- Retrieve Journal Entries (QjoRetrieveJournalEntries) API

«

Considerations for entries which contain minimized entry-specific data: You can reduce the size of journal receivers by specifying minimized entry-specific data on the Create Journal (CRTJRN) and Change Journal (CHGJRN) commands.

If you have selected to use the MINENTDTA parameter for the journal, then some of your journal entries entry-specific data will be minimized. The entries will only be minimized if the minimization technique will deposit a journal entry which is smaller in size than the complete entry would be. Use the Journal entry information finder to see which specific journal entry types can possibly be minimized. When the entry is minimized, the fixed-length portion of the journal entry will have the minimized entry-specific data indicator on. Currently, only data areas and database physical files can have their entry-specific data minimized.

Data area considerations

The layout of the data area entries which are minimized is exactly the same as the layout if the entry was not minimized. The only difference is that only the bytes which actually changed are deposited rather than depositing all the bytes on the change request. See Update data area journal entries for the entry layout of the change data area entries.

Database physical file considerations

The layout of the minimized record changes entries is completely different than the layout when the entry is not minimized. The data it not even recognizable nor readable as sophisticated hash techniques are used in addition to only operating on actual changed bytes. Additionally, the Null-value-indicators field will be used, even if the file is not null capable, to provide additional information that can be used by database operations. Therefore, if you want to use the journal as an audit mechanism, you may not want to choose this option for database physical files since you will not be able to read the actual change made.

Remote journal management

Remote journal management allows you to establish journals and journal receivers on a remote system or to establish journal and receivers on independent disk pools that are associated with specific journals and journal receivers on a local system. The remote journaling function can replicate journal entries from the local system to the journals and journal receivers that are located on the remote system or independent disk pools after they have been established.

Use the following information to set up remote journal management:

- Remote journal concepts
- Plan for remote journals
- Set up remote journals
- Remove remote journals
- Activate and inactivate remote journals
- Manage remote journals
- Scenarios: Remote journal management and recovery

Remote journal concepts

Remote journal management helps to efficiently replicate journal entries to one or more systems. You can use remote journal management with application programs to maintain a **data replica**. A data replica is a copy of the **original data** that resides on another iSeries(TM) server or independent disk pool. The original data resides on a primary system. Applications make changes to the original data during normal operations.

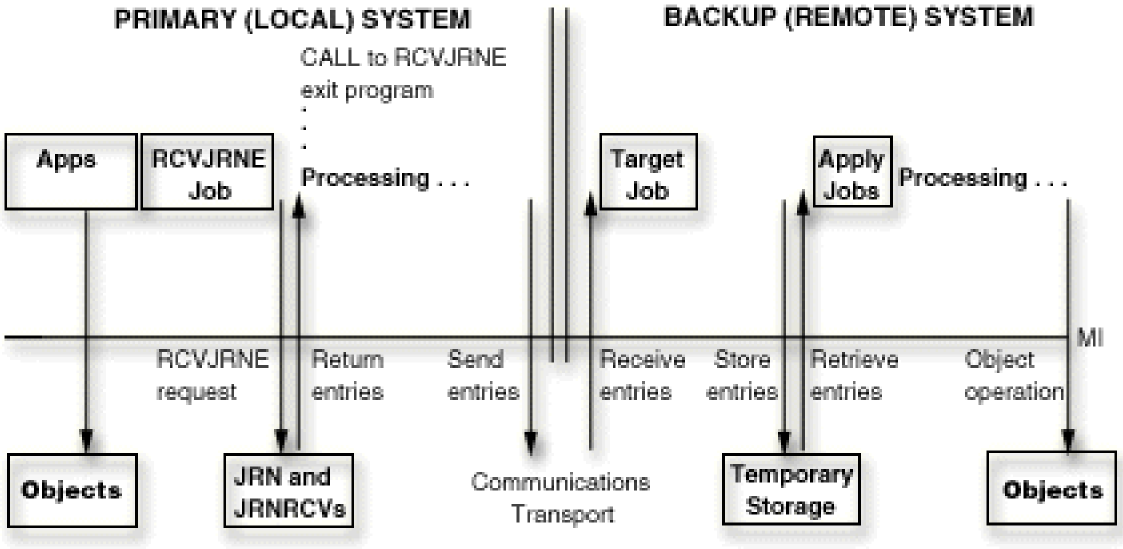
Prior to V4R2M0, you could have accomplished a similar function by using the Receive Journal Entry (RCVJRNE) command. In that environment, the RCVJRNE exit program receives journal entries from a journal, and then sends the journal entries to the remote system by using whatever communications method is available. All of this processing occurs asynchronously to the operation that is causing the journal entry deposit and takes place at an application layer of the system.

The remote journal function, however, replicates journal entries to the remote system at the Licensed Internal Code layer. Moving the replication to this lower layer provides the following:

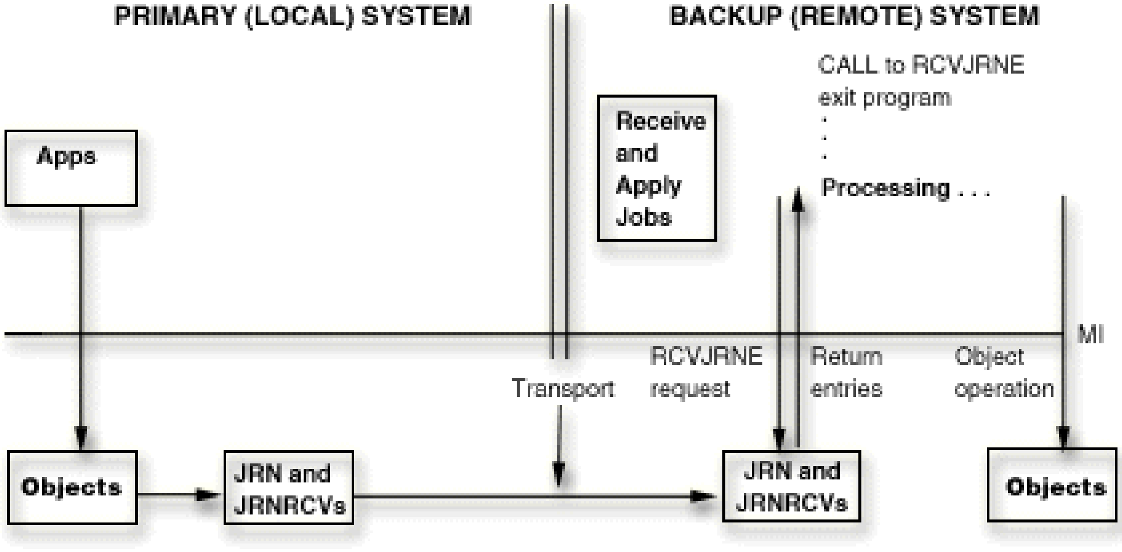
- The remote system handles more of the replication overhead
- Overall system performance and journal entry replication performance is improved
- Replication can (optionally) occur synchronously to the operation that is causing the journal entry deposit
- Journal receiver save operations can be moved to the remote system.

The figures below illustrate a comparison of a hot-backup environment with and without remote journal management. **Hot-backup** is the function of replicating an application's dependent data from a **primary system** to a **backup system**. The primary system is the system where the original data resides. The backup system is the system where a replica of the original data is being maintained. In the event of a primary system failure, you can perform a switch-over to the backup system.

Hot-backup environment without remote journal function, and application-code based apply



Hot-backup environment with remote journal function, and application-code based apply

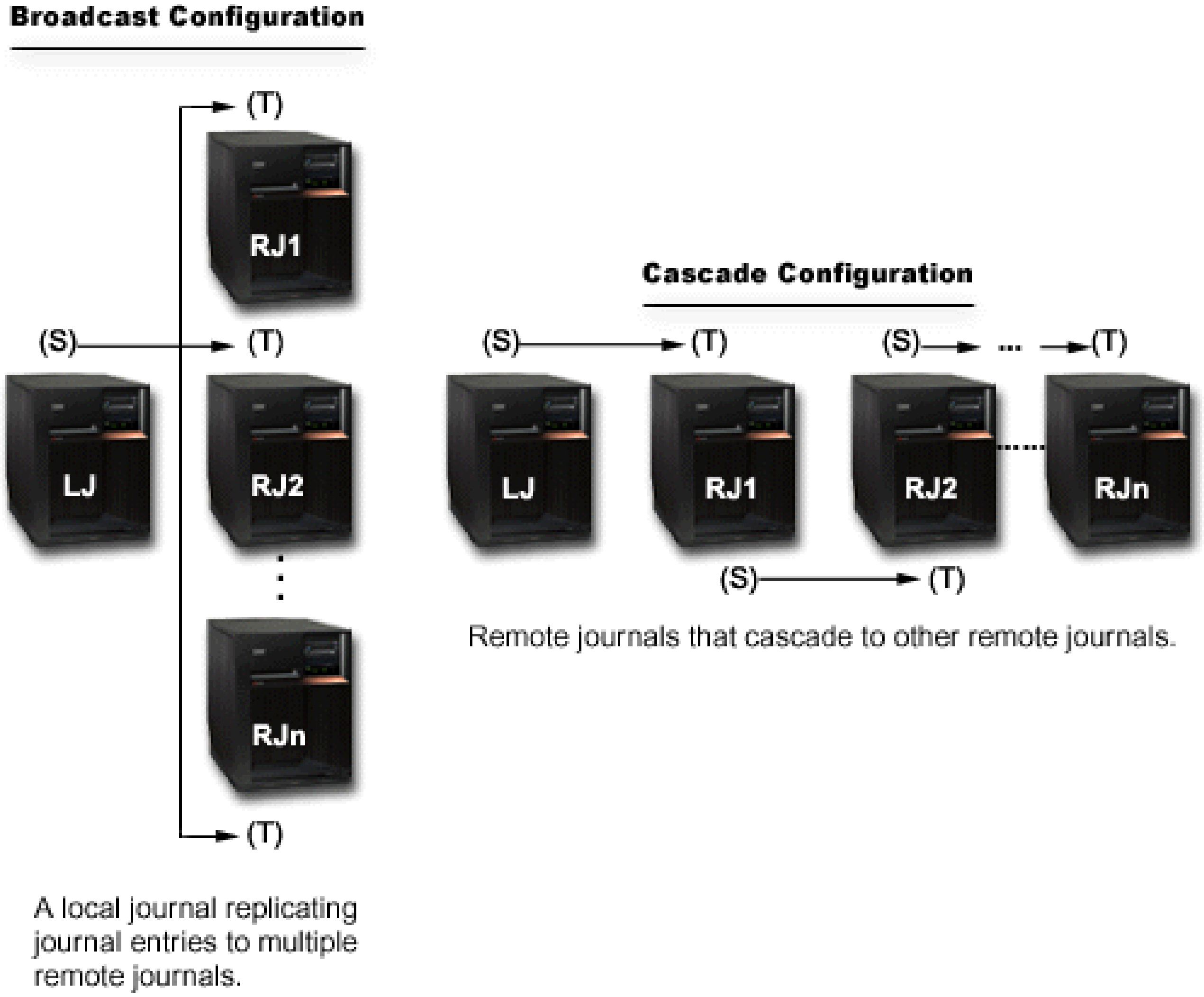


The following topics provide more information about remote journaling:

- Network configurations for remote journals
- Types of remote journals
- Journal state and delivery mode
- Journal receivers associated with a remote journal
- Add remote journal process
- Supported communications protocols for remote journals
- Release-to-release considerations for remote journals

Network configurations for remote journals

The following figure shows the two basic remote journal function configurations.


Broadcast Configuration
(T)
RJ1
(S)
(T)
LJ
RJ2
RJn
(T)
A local journal replicating journal entries to multiple remote journals.
Cascade Configuration
(S)
(T)
(S)
...
(T)
LJ
RJ1
RJ2
RJn
(S)
(T)
Remote journals that cascade to other remote journals.

A **broadcast** configuration is a journal that replicates its journal entries to one or more remote journals. A **cascade** configuration is a remote journal that replicates its journal entries to an additional remote journal. The additional remote journal can replicate the entries to yet another remote journal, and so on. The remote journal function configurations can stand alone or can be combined with one another. For example, one or more of the remote journals in the broadcast configuration could cascade down to several additional remote journals. Likewise, one or more remote journals in the cascade configuration could broadcast out to one or more remote journals.

A **local journal** is populated by applications that are depositing journal entries. A **remote journal** is populated by receiving its journal entries from either a local or another remote journal. The journals are paired, as depicted in the preceding figure where (S) represents a journal on a source system, and (T) represents a journal on a target system. In the cascade configuration, a remote journal can be a recipient of journal entries (a target), and a replicator of journal entries (a source) at the same time.

A **source system** is a system where a journal resides and is having its journal entries replicated to a remote journal on a target system.

Note: A source system is not necessarily the primary system. For example, a remote journal that is cascading its journal entries to another remote journal is said to reside on a source system.

A **target system** is a system where a remote journal resides and is receiving journal entries from a journal on a source system.

A **remote journal network** includes the local journal and all of the remote journals that are downstream from that local journal. You can set up the remote journal network in broadcast configuration, cascade configuration, or a combination of the two configurations.

In many environments, users attempt to minimize the amount of processing that the local or primary system performs by shifting as much of the processing as possible to other systems in the network. A combination of the broadcast and cascade configurations allows for this when replicating the journal entries from a single system to multiple other systems. For example, replicating a local journal to a single remote journal on a target system will minimize the replication cost on the primary system. Then, from the target system, the replicated journal can be asynchronously replicated by either a broadcast or cascade configuration to other remote journals on other systems. This allows all of the journal entries to be known to all desired systems, while requiring a minimal amount of processing on the primary system.

The following characteristics apply to local journals and to any journal receivers that were attached to local journals:

- Objects can be journaled to local journals.
- Journal entries can be directly deposited to local journals. For example, the Send Journal Entry (SNDJRNE) command or the Send Journal Entry (QJOSJRNE) API can be used to send journal entries directly to a local journal.

The following characteristics apply to remote journals and to any journal receivers that were attached to remote journals:

- Objects cannot be journaled to remote journals.
- Journal entries cannot be directly deposited to remote journals. For example, the Send Journal Entry (SNDJRNE) command or API (QJOSJRNE) cannot be used to send journal entries directly to a remote journal.
- Journal entries are only replicated to remote journals from an associated **source journal**. A source journal is the journal on the source system to which a remote journal has been added. A source journal can be either a local or a remote journal.
- The information in the journal entries such as time stamps, system name, and qualified journal receiver names reflect information as deposited in the local journal for this remote journal network.
- The information in the journal receiver such as attach time and detach time reflect the information as it is for the local journal for the remote journal network.
- Certain attributes of the remote journal are fixed and determined based on the source journal, such as the values for receiver size options and the values for minimize entry-specific data. These attributes for the remote journal cannot be changed except by changing the attributes for the source journal.

Types of remote journals

The two types of remote journals are *TYPE1 and *TYPE2. The two types identify operational characteristics of a remote journal and its associated journal receivers. The following table is an overview of the different remote journal types and their characteristics. There are no performance differences between the types of remote journals.

	Local journal	***TYPE1 remote journal**	***TYPE2 remote journal**
Remote journal types that can be added	*TYPE1 *TYPE2	*TYPE1 *TYPE2	*TYPE2
Remote journal name	N/A	Journal name must be the same as the local journal.	Journal name may be different from the source journal.
Journal library redirection	N/A	Journal library name may be redirected to a single different library from that of the local journal. All *TYPE1 remote journals associated with a given local journal must reside in the same named library.	A given redirected library may be specified when adding a remote journal. Subsequent adds of *TYPE2 remote journals may specify a different library redirection than was specified on any previously added remote journal.
Journal receiver library redirection	N/A	Receiver library name may be redirected to a single different library from that of the receivers associated with the local journal.	A given redirected library may be specified when adding a remote journal. Subsequent adds of *TYPE2 remote journals may specify a different library redirection than was specified on any previously added remote journal.
Journal receiver library redirection used on activate	N/A	The target library used when replicating a receiver from the source journal to this remote journal will reflect the library redirection that was in place for the receiver, if any, at the time the receiver was attached to the source journal.[1]	The target library used when replicating a receiver from the source journal to this remote journal will reflect the library redirection that is currently defined for the target journal.
Receiver restore characteristics[2, 3]	Receivers associated with the local journal can be saved and restored to the local system or to any of the systems for the *TYPE1 remote journals and be linked into the correct receiver chain of the local journal or the *TYPE1 remote journal.	Receivers associated with the local journal or any of the *TYPE1 remote journals can be saved and restored to the local system or to any of the systems where the *TYPE1 remote journals reside and be linked into the correct receiver chain of the journal.	Receivers associated with a given *TYPE2 remote journal can be saved and restored to the local system or to the same system where the *TYPE2 remote journal resides and be linked into the correct receiver chain of the journal.

	Local journal	*TYPE1 remote journal	*TYPE2 remote journal

Notes: [1]If the journal receiver was attached to a journal when no remote journals were added, then no library redirection is assumed for that journal receiver if that receiver is specified during activation. Therefore, the journal receiver will be created in the same library on the target system as it is on the local system.

[2]A journal receiver from any system in the remote journal network may always be restored to any system if the receiver is being restored into the original or redirected receiver library. Otherwise, receivers can always be restored to any system and associated with a local journal if a local journal by the same name as the original local journal is found residing in the same named original local journal library.

[3]If a journal receiver's original or redirected library exists in an independent disk pool, then the ASP group name for the independent disk pool is used in place of the system name when making restore decisions.

See Considerations for save and restore operations with remote journals for more information.

Journal state and delivery mode

The **journal state** describes an attribute for a journal. The attribute value can be *ACTIVE, *INACTIVE (remote journal only), or *STANDBY (local journal only). For a local journal, *ACTIVE indicates that journal entries are currently allowed to be deposited into the journal. *STANDBY indicates that most journal entries are not deposited.

You can view the journal state for a remote journal on a target system that is associated with a journal on a source system in one of two ways:

- When viewed from the source system, *ACTIVE indicates that journal entries are currently being replicated to that remote journal on the target system. *INACTIVE indicates that journal entries are not currently being replicated.
- When viewed from the target system, *ACTIVE indicates that journal entries are currently being received from the journal on the source system. *INACTIVE indicates that the target journal is not ready to receive journal entries from the source journal.

The following table provides a summary of the journal type, delivery mode and journal state interactions.

Journal type	Delivery mode	Journal state	Comments
*LOCAL	Not applicable	*ACTIVE	Objects journaled to the local journal can be changed, and entries can also be deposited into the local journal using the Send Journal Entry (SNDJRNE) command or the Send Journal Entry (QJOSJRNE) API interfaces. The currently attached journal receiver may or may not be currently replicated to one or more remote journals. This depends upon whether any remote journals have been added to the local journal's definition, and if so, the current journal state for each of those remote journals.

Journal type	Delivery mode	Journal state	Comments
*LOCAL	Not applicable	*STANDBY	This is the state of a local journal after the Change Journal (CHGJRN) command specifying JRNSTATE(*STANDBY) is used to not allow deposits into the local journal. The local can journal can also be in *STANDBY state after an IPL if the local journal is in *STANDBY state when the system ends. Objects journaled to the local journal **can** be restored or changed, but most journal entries are not deposited until the journal state for the local journal is again changed to *ACTIVE. This can be performed by using the Change Journal (CHGJRN) command specifying JRNSTATE(*ACTIVE).
*REMOTE	*SYNCPEND	*ACTIVE	This is the state after a remote journal has been activated using the Change Journal State (QjoChangeJournalState) API or CHGRMTJRN command and the processing is still in the catch-up phase of remote journal activation. Synchronous delivery mode was requested on the API invocation.
*REMOTE	*SYNC	*ACTIVE	This is the state after a remote journal has been activated using the Change Journal State (QjoChangeJournalState) API or CHGRMTJRN command, after catch-up has completed, and changes to the currently attached journal receiver for the journal on the source system are being replicated synchronously to the remote journal on the target system.

Journal type	Delivery mode	Journal state	Comments
*REMOTE	*ASYNCPEND	*ACTIVE	This is the state after a remote journal has been activated using the Change Journal State (QjoChangeJournalState) API or CHGRMTJRN command and the processing is still in the catch-up phase of remote journal activation. Asynchronous delivery mode was requested on the API invocation.
*REMOTE	*ASYNC	*ACTIVE	This is the state after a remote journal has been activated using the Change Journal State (QjoChangeJournalState) API or CHGRMTJRN command, after catch-up has completed, and changes to the currently attached journal receiver for the journal on the source system are being replicated asynchronously to the remote journal on the target system.
*REMOTE	*SYNC	*INACTPEND	This is the state of a remote journal, viewed from the target system where some failure has occurred and either the system is in the process of inactivating the remote journal, or unconfirmed journal entries exist in the remote journal. See Confirmed and unconfirmed journal entries for more information.
*REMOTE	*ASYNC	*INACTPEND	This is the state of a remote journal, viewed from the target system where some failure has occurred and the system is in the process of inactivating the remote journal.

Journal type	Delivery mode	Journal state	Comments
*REMOTE	*ASYNC	*CTLINACT	This is the state after a remote journal has been made inactive using the Change Journal State (QjoChangeJournalState) API or CHGRMTJRN command, a controlled deactivate was requested on that call and that controlled deactive has not yet completed.
*REMOTE	Not applicable	*INACTIVE	This is the state after a remote journal has been added and associated with a journal on a source system. However, the journal state for the added remote journal has yet to be activated or has been made inactive using the Change Journal State (QjoChangeJournalState) API, CHGRMTJRN command, or by an IPL. No delivery mode is in effect for an inactive remote journal.

Journal receivers associated with a remote journal

Journal receivers that are associated with a remote journal are exact replicas of the corresponding journal receivers that are associated with the journal on the source system. The receiver directory for a remote journal is maintained in the same way as the receiver directory is maintained for the related source journal. Consecutive receivers associated with a remote journal are linked together to form a receiver chain. Receiver chain breaks are forced and maintained in a similar manner for local and remote journals.

However, the following are some other differences for remote journals and the journal receivers that were attached to remote journals:

- A remote journal does not have to have a currently attached journal receiver. However, if the remote journal is ready to receive journal entries, then it must have an attached receiver; all the journal entries will be replicated to that attached receiver.
- The receiver that is currently attached to a remote journal that is in the catch-up phase can be a different journal receiver than is currently attached to the source journal.
- The receiver that is currently attached to an asynchronously maintained remote journal can be a different journal receiver than is currently attached to the source journal.
- The receiver that is currently attached to a synchronously maintained remote journal is the same journal receiver as is currently attached to the source journal.
- You can delete the journal receiver that is attached to a remote journal if the journal state of that journal is not *ACTIVE.
- You can delete the journal receivers that are associated with a remote journal in any order, regardless of their position within the receiver directory chain.
- The creation date and time stamps for remote journals are always those of the system on which the journals were created by the remote journal function. This is also true for journal receivers that were attached to remote journals.

- The save and restore date and time stamps for remote journals are always those of the system on which the save or restore operation took place. This is also true for the journal receivers that are associated with the remote journals.
- The attach and detach time stamps for a journal receiver that was attached to a remote journal are always those of the attach and detach time stamps of the local journal receiver.
- When a journal receiver that is associated with a remote journal is saved, deleted or restored, the following journal entries are not deposited:
  - J RD - Journal receiver deleted
  - J RF - Journal receiver saved, storage freed
  - J RR - Journal receiver restored
  - J RS - Journal receiver saved

For more information about journal receiver directory chains, see Keep track of journal receiver chains.

Add remote journal process

Adding a remote journal creates a remote journal on a target system or independent disk pool and associates that remote journal with the journal on the source system. This occurs if this is the first time the remote journal is being established for a journal. The journal on the source system can be either a local or remote journal.

If a remote journal environment has previously been established, adding a remote journal re-associates the remote journal on the target system with the journal on the source system.

You can establish and associate a remote journal on a target system with a journal on the source system by one of the following methods:

- iSeries(TM) Navigator.
- Add Remote Journal (QjoAddRemoteJournal) API on the source system.
- Add Remote Journal (ADDRMTJRN) command on the source system.

What happens during add remote journal processing

Some of the processing which takes place as part of adding a remote journal is as follows:

- A check is performed on the target system to verify that the user profile adding the remote journal exists. A user profile with the same name as the user profile which is adding a remote journal must exist on the target system. If the profile does not exist on the target system, then an exception is signaled, and the processing ends.
- A check is performed to verify that the target system has a library by the same name as the library for the journal on the source system. If the library does not exist on the target system, then an exception is signaled, and the processing ends.
- A check is performed on the target system to determine if a journal by the same qualified name as the journal on the source system already exists. If a journal already exists, it can be used for the remainder of the add remote journal processing if it meets the following conditions:
  1. It is a remote journal.
  2. It was previously associated with this same source journal or part of the same remote journal network.
  3. The type of the remote journal matches the specified remote journal type.
- If a journal was found, but does not meet the above criteria, then an exception is signaled, and the processing ends. Otherwise, the remote journal is used for the rest of the add remote journal processing.
- If no journal is found on the specified target system, then a remote journal is created on the target system. The new remote journal has the same configuration, authority, and audit characteristics of the source journal. The journal that is created has a journal type of *REMOTE.

The creation of the journal on the target system is performed as though the journal was being saved and restored to the target system. Therefore, the ownership of the journal on a target system will follow the same rules as with the existing save and restore functions. If the user profile which owns the journal on the source system is on the target system, then that profile will own the created journal on the target system. If the user profile does not exist on the target system, then the profile QDFTOWN will own the journal on the target system.

Additionally, if the remote journal is created, the values for the journal attributes of text, journal message queue, delete receivers value, and delete receiver delay time will be taken from what is specified on the API invocation. After the remote journal has been created, these values can be changed by using the Change Journal (CHGJRN) command for the remote journal on the remote system. After the remote journal is created, any changes to these attributes on the source journal will not cause equivalent changes to the remote journal. See Remote journal attributes for more information.

When adding the remote journal, you must specify the type of remote journal to add. The remote journal type influences the library redirection rules and other operational characteristics for the journal. See Types of remote journals for more information.

Guidelines for adding a remote journal

The following are guidelines for adding a remote journal:

- You can only associate a remote journal with a single source journal.

Note: The same remote journal can then have additional remote journals that are associated with it that are located on other target systems. This is the cascade configuration that is shown in Network configurations for remote journals.

- The remote journal will only have its attached receiver populated with journal entries that are replicated from the corresponding journal receiver on the source system. No journal entries can be directly deposited to a remote journal.
- A maximum of 255 remote journals can be associated with a single journal on a source system. This can be any combination of asynchronously maintained or synchronously maintained remote journals.

Synchronous and asynchronous delivery mode has more information. Library redirection with remote journals and Remote journal attributes provide more concepts about the add remote journal process. Add remote journals provides the steps for adding a remote journal.

Library redirection with remote journals: **Library redirection** provides a means for remote journals and any of their associated journal receivers to optionally reside in differently named libraries on the target system from the corresponding local journal and journal receivers on the local system. You can specify library redirection by using one of the following:

- iSeries(TM) Navigator
- Add Remote Journal (QjoAddRemoteJournal) API
- Add Remote Journal (ADDRMTJRN) command

When using the QjoAddRemoteJournal API, specify a different name in the Remote Journal Library name field or the Remote Journal Receiver Library field. When using the ADDRMTJRN command, specify a different name for the Target Journal Library parameter or the Remote Receiver Library parameter. When a remote journal is added, its journal type specification influences how much redirection you can specify.

Types of remote journals describes the various types of remote journals that can be added, as well as a description of their redirection characteristics.

If redirection is not specified, then the remote journal will reside in a library that has the same name as the library that contains the source journal.

Note: Library redirection for the journal object must be specified when replicating the journal entries to a target system for any journal starting with the letter Q in a library starting with Q. This does not apply to the QGPL library. This restriction prevents collisions between local and remote journals that are used for system functions. One example of this is journal QAUDJRN in library QSYS which is used for security auditing.

If no redirection is specified for the journal receiver, then the remote journal receiver will reside in a library whose name is the same as the library for the source journal receiver. For example, the source journal has two receivers that are associated with it, receiver RCV0001 in library LIBA, and receiver RCV0002 in library LIBB. If no journal receiver library redirection is specified, then the journal entries in RCV0001 in library LIBA on the source will be replicated to RCV0001 in library LIBA on the target system. The journal entries in RCV0002 in library LIBB on the source will be replicated to RCV0002 in library LIBB on the target system. Therefore, both libraries, LIBA and LIBB, will need to exist on the target system prior to the invocation of the remote journal function. If journal receiver library redirection is specified with a redirected receiver library specification of RMTLIB, then both RCV0001 and RCV0002 would be in library RMTLIB on the target system.

For *TYPE1 remote journals, the library redirection or the selection of no library redirection for the journal and journal receivers can only be modified by doing the following:

- Remove all *TYPE1 remote journals.
- Change the local journal and attach a new journal receiver.
- Delete the remote journal from the target system.
- Add the *TYPE1 remote journal, specifying the new library redirection, if any.

For *TYPE2 remote journals, the library redirection or the selection of no library redirection for the journal and journal receivers can only be modified by doing the following:

- Remove the *TYPE2 remote journal.
- Delete the remote journal from the target system.
- Add the *TYPE2 remote journal, specifying the new library redirection, if any.

Independent disk pools and library redirection

If you want the remote journal on an independent disk pool on the target system, specify a library on the target system that is on an independent disk pool for that system and specify an RDB entry for the independent disk pool.

If you place your remote journal on an independent disk pool on the target system, the following rules apply:

- The independent disk pool on the target system must be varied on.
- The independent disk pool must be a library capable disk pool.
- The remote journal, the remote journal receiver, and the message queue must be in the same independent disk pool group.

See Journal management and independent disk pools for information about independent disk pools as they relate to journaling.

Remote journal attributes: When a remote journal is created by the add remote journal processing, the remote journal's initial attributes are defined by the add request and the source journal. Various journal attributes for a remote journal are treated as follows:

Disk pool
If the library for the remote journal resides in a disk pool, the remote journal will be created in that disk pool.

Journal message queue
Defined on add request. Once the remote journal is created, the journal message queue, can be modified by using the Change Journal (CHGJRN) command on the remote journal on the remote system.

Delete receivers
Defined on add request. Once the remote journal is created, the delete receivers attribute can be modified by using the CHGJRN command on the remote journal on the remote system.

Manage receivers
Does not apply. The managing of the receivers for the remote journal is driven by the management of the source journal.

Minimize entry-specific data options
Does not apply. The minimize entry-specific data options in effect for the remote journal are driven by the minimize entry-specific data options in effect for the local journal.

Receiver size options
Does not apply. The receiver size options in effect for the remote journal are driven by the receiver size options in effect for the source journal.

Text
Defined on add request. Once the remote journal is created, the text can be modified by using the CHGJRN command on the remote journal on the remote system.

Manage receiver delay
The managing of the receivers for the remote journal is determined by the management of the source journal.

Delete receiver delay
Defined on add request. Once the remote journal is created, the delete receiver delay attribute can be modified by using the CHGJRN command on the remote journal on the remote system.

Fixed-length data
Does not apply. The fixed-length data options in effect for the remote journal are driven by the fixed-length data options in effect for the local journal

Journal cache
Does not apply.

Supported communications protocols for remote journals

The remote journal function supports the following communications protocols for replicating the journal entries to the remote systems:

- **OptiConnect for OS/400^(R)**. If you want to use the OptiConnect for OS/400 support, you must purchase and install the required hardware and software for that support. Refer to OptiConnect for OS/400 for more information.

- **Systems Network Architecture (SNA)**. If you want to use SNA for the transport, there are no additional software considerations. The software support is in the base operating system. You must purchase whatever hardware is appropriate for your configuration. For more information see SNA Distribution Services on the V5R1 Supplemental Manuals Web site.
- **Transmission Control Protocol/Internet Protocol(TCP/IP)**. If you want to use TCP/IP for the transport, there are no additional software considerations. The software support is in the base operating system. You must purchase whatever hardware is appropriate for your configuration. Refer to TCP/IP Configuration and Reference for more information.
  » **Note:** All remote journal TCP connections use the TCP local port of 3777.«

Specifying a relational database (RDB) directory entry will identify the communications protocol that the remote journal function will use. The RDB that is specified must meet the following rules:

- The communications protocol must be one of the remote journal function supported protocols.
- The remote location name in the RDB cannot refer to the *LOCAL database.
- The RDB cannot use an application requester driver program (*ARDPGM) to locate the target system.

For more information about creating relational databases, refer to Distributed Database Programming

Security of the remote journal function is dependent on the communications protocol security. The remote journal function does not alter the security characteristics that are available.

The communications function that is identified by the RDB can be shared by other activity. However, you may consider isolating the remote journal function activity in order to have the best performance.

Release-to-release considerations for remote journals

Release-to-release considerations for remote journals are as follows:

- Information APAR II12556 contains the same list of program temporary fixes (PTF) for V5R1.
- If you specify RCVSIZOPT(*MAXOPT2) on the journal that you attach a journal receiver to, you cannot replicate the journal receivers to any remote journals on any systems at a release prior to V5R1M0.
- If you specify minimized-entry specific data (MINENTDTA) for either *FILE or *DTAARA on the journal to which you attached a journal receiver, you cannot replicate the journal receivers to any remote journals on any systems at a release before V5R1M0.
- »If you specify RCVSIZOPT(*MAXOPT3) on the journal that you attach a journal receiver to, you cannot replicate the journal receivers to any remote journals on any systems at a release prior to V5R3M0.«

Plan for remote journals

The following topics provide detailed information for planning to set up remote journals:

- Journals that are good candidates for remote journal management
- Synchronous and asynchronous delivery mode for remote journals
- Communications protocol and delivery mode for remote journals
- Where the replication of journal entries start
- Factors that affect remote journal performance
- Remote journals and auxiliary storage
- Journal receiver disk pool considerations
- Remote journals and main storage

Journals that are good candidates for remote journal management

Journals that you are currently replicating, or that you plan to replicate, in their entirety to one or more systems, are excellent candidates for the remote journal function.

Journals with high activity that require frequent saves and deletes of the associated journal receivers during the day are also good candidates for the remote journal function. If you use remote journaling, you can specify that the backup system takes over the journal receiver save processing. Then the primary system can specify system journal-receiver management and automatic deletion of journal receivers. This frees up disk space on the primary system as quickly as possible. The backup system is the system where a replica of the original data is being maintained. The primary system is the system where the original data resides.

Also, you might have applications that are so critical to your business that any downtime will impact your operations. The **application dependent data** is a good candidate to protect with the remote journal function. Application dependent data is any data that a particular application depends on if that application is interrupted and has to be restarted.

For example, you may have a database that has a lot of query activity that impacts your system performance. That local database is a good candidate to replicate to another system so that the query activity moves from the local system to that remote system. The remote journal function can assist this process of replicating the database.

Synchronous and asynchronous delivery mode

The terms **asynchronously maintained** and **synchronously maintained** both describe a remote journal function delivery mode for journal entry replication. If a journal is asynchronously maintained, control is returned to the application generating the journal entry on the source system without waiting for the journal entry to be replicated to the remote journal. An asynchronously maintained remote journal might lag several journal entries behind the total number of journal entries in the journal on the source system.

If a journal is synchronously maintained, control is not returned to the application generating the journal entry on the local system until the journal entry is replicated to the remote journal.

Synchronous delivery mode

Synchronous delivery means that the journal entry is replicated to the target system concurrently with the entry being written to the local receiver on the source system. The entry is known on the target system, in main storage, prior to returning control to the user application that deposited the journal entry on the source system. Therefore, the target system knows of all journal entries as they are being made in real-time on the source system. Using this mode allows for recovery without losing journal entries on the target system if the source system fails. Providing journal entries synchronously to a target system will have some affect to the journaling throughput on the local system.

Synchronous delivery mode is only supported when a remote journal is associated with a local journal.

»

There are certain circumstances, when using synchronous mode, when some journal entries are not immediately sent to the target system. These entries are either not necessary for recovery or the user did not specify that they be forced to disk. Journal entries are sent to the remote journal at the same time that they are forced to disk for the source journal. Since these entries are not forced to disk at deposit time, they are not sent to the remote system.

«

- Some entries that are not required for data recovery might not be immediately sent to the target system. For example, journal entries for a file close (journal code ’F’, entry type ’CL’) or a stream file open, (journal code ’B’, entry type ’OF’).
- User-generated journal entries that use the Send Journal Entry (SNDJRNE) command or the Send Journal Entry API (QJOSJRNE) might not be sent to the target system. If either you, or your

application, do not request to force these user-generated entries they will only be replicated to the remote journal when some other action forces them. Therefore, periodically specify FORCE(*YES) when using the send journal entry functions.

- Journal entries that are associated with commitment control transactions might not be immediately sent to the remote system. These entries will be retrievable after the following journal entries have been deposited into the source journal:
  - Journal code ’C’, journal entry type ’CM’ (Commit)
  - Journal code ’C’, journal entry type ’RB’ (Rollback)

  See Remote journal considerations for retrieving journal entries when using commitment control for more information. When journal caching is being used (JRNCACHE(*YES) on the CHGJRN command), entries that exist only in the cache are not available on the target system. With journal caching, entries are not sent to the target system until they are written from the cache to disk on the source system.
- If the local journal is using journal caching, then journal entries will be bundled up before they are sent to the target.

Asynchronous delivery mode

Replicating a journal entry **asynchronously** means that the journal entry is replicated to the target system after control is returned to the application depositing the journal entry on the source system. Using this mode allows for recovery that might lose some journal entries if the source system fails. However, this mode has less affect to the journal throughput on the local system in comparison with the synchronous mode.

Journal entry latency might occur when remote journals are asynchronously maintained. Journal entry latency is the difference between the journal entries that exist in the remote journal on the target system from those residing in the journal on the source system. From a recovery standpoint, the source system might be some number of journal entries ahead of what journal entries are known on the target system.

Communications protocol and delivery mode for remote journals

The greater the volume of traffic, that is the higher the rate of journal entry deposits, the faster communications method you must choose. If your traffic is minimal, then a slower communications method can be adequate.

The **delivery mode** defines how journal entries are replicated to a remote journal. The delivery mode only applies when actively replicating the journal entries from a journal on a source system to a remote journal on a target system. The delivery mode can be either synchronous or asynchronous.

If the application dependent data is critical and the loss of journal entries can impact your business, then use the synchronous delivery mode. Synchronous delivery mode is only valid when activating a remote journal that is associated with a local journal.

It may be acceptable that the remote system does not have all the journal entries as they are being deposited or replicated into the source journal. If this is true, the asynchronous delivery mode is a good choice to minimize the impact to the source journaling throughput.

The choice of delivery mode and communications protocol are closely linked. Since the synchronous delivery mode will affect the interactive users response time, the faster the communications protocol the better. This again will be dependent on the journal entry deposit rate.

Where the replication of journal entries start

When you specify a journal receiver for remote journaling, you are specifying where the replication of journal entries will start. You can choose from the following options:

Attached receiver on target system

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.

The remote journal on the target system might not have an attached journal receiver. If this occurs, the journal receiver that is currently attached to the journal on the source system is created on the target system. That journal receiver is then 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, no journal entries can be replicated, and an error is returned. This is only possible in the case of a remote journal that is associated with another remote journal.

To use this option specify one of the following:

- *ATTACHED value on the CHGRMTJRN command.
- **Use attached receiver on target system; otherwise, on source system** in the **Activate** dialog in iSeries(TM) Navigator.

Attached receiver on source system only

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. Replication starts 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. The system returns an error.

If the corresponding journal receiver does not exist on the target system, the journal receiver is created and attached to the remote journal on the target system. Journal entries then 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, journal entries cannot be replicated, and the system returns an error. This is only possible in the case of a remote journal that is associated with another remote journal.

To use this option specify one of the following:

- *SRCSYS value on the CHGRMTJRN command.
- **Use attached receiver on source system only** in the **Activate** dialog in iSeries Navigator.

Qualified journal receiver name

The replication of journal entries starts with the specified journal receiver name for 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. Replication starts 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. The system returns an error.

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

If the journal on the source system is not associated with the specified journal receiver, no journal entries can be replicated, and an error is returned.

The creation of any receiver on the target system by the change journal state processing is performed as though the receiver was being saved and restored to the target system. Therefore, the ownership of the receiver on a target system will follow the same rules as with the existing save and restore functions. If the user profile which owns the receiver on the source system is on the target system, then that profile will own the created receiver on the target system. If the user profile does not exist on the target system, then the profile QDFTOWN will own the receiver on the target system.

Additionally, information such as the audit attributes of the source journal receiver at the time it was attached to the source journal will be incorporated into the created journal receiver on the target system.

If the library for the journal receiver resides in an ASP, the journal receiver will be created in that ASP. The remote journal function does not support nonlibrary ASPs for the ASP of the remote journal receiver. See Journal receiver disk pool considerations for more information.

Factors that affect remote journal performance

»

There are two main performance objectives for the remote journal function. To provide a timely delivery of journal entries to a target system and to minimize impacts to the journaling throughput on the source system. Even though both aspects are very important for both synchronous and asynchronous delivery modes, each mode prioritizes the two in a different order. The top priority for synchronous delivery is to guarantee that the remote journal is always up to date with the source journal. For asynchronous delivery mode, the top priority is to minimize impacts to journaling throughput.

«

All performance considerations that are currently used for a local journal still apply and must continue to be employed. The following are additional factors that may affect the performance of the remote journal function. The factors are listed in the order of importance.

1. **Transport method** »

   Your choice of transport depends on the rate of the journal activity in your environment. Make special consideration for using a fast transport method when you use synchronous delivery mode. Weigh the response time impacts of the synchronous delivery mode in your environment against the communications overhead of the transport method you choose.

   When replicating journal entries over a long distance, the most important performance factors regarding a communications transport method are the overall rated speed of the communications resource and any existing traffic already using the communications resource.

   For more information about transport methods, see the Networking topic.«

2. **Number of remote journals that are being maintained**

   With respect to the job performing the journal entry deposit, the impact of the remote journal function for asynchronously maintained journals is not noticeable. For synchronously maintained journals, the impact depends on the slowest connection rather than number of remote journals.

   The impact to the job performing the journal entry deposit for an asynchronously maintained journal is significantly less than that for a synchronously maintained journal. Also, it is recommended that only one synchronous remote journal be maintained for a given local journal.

   With respect to the system performance impacts, the processor use typically increases by less than an equal factor for each additional remote journal.

3. **Arrival rate of journal entries that are being deposited on the local system**

   The higher the arrival rate of journal entries being deposited on the local system, the greater the chance journaling throughput will increase for synchronous or asynchronous delivery. A high arrival rate might cause asynchronous journaling to fall further behind.
4. **Batch versus Interactive**

   In general, higher local and remote journal throughput can be maintained when many interactive jobs generate the journal throughput rather than a single-threaded batch job. Journal caching can also increase this throughput for batch processing regardless of the number of jobs.
5. **Processor utilization on the source system**

   The higher the processor utilization of the source system, the greater the chance of affecting journaling throughput for synchronous or asynchronous delivery. This may cause asynchronous journaling to fall further behind.
6. **Processor utilization on the target system**

   The higher the processor utilization of the target system, the greater the chance of affecting journaling throughput for synchronous or asynchronous delivery. This may cause asynchronous journaling to fall further behind.
7. **The value set for the sending task priority when using the asynchronous delivery mode**

   The larger the value, the smaller effect the remote journal function will have on the system, but the further the target system may lag behind the source system.

Performance considerations regarding the catch-up phase when activating the remote journal function include the following in order of importance:

Note: The catch-up processing that is performed by the remote journal function is the most efficient method of replicating the journal entries with the remote journal function.

1. **Total number of bytes for all of the journal entries that need to be caught up**

   The larger the total size, the longer the catch-up phase will run.
2. **Transport method**

   » Select a transport method that is appropriate for your remote journaling environment. «
3. **Disk protection on the target system**

   At high data transfer rates, disk units with device parity protection in the ASP on the target system can limit the performance of the catch-up phase, unless the target system has sufficient write cache configured in the I/O adaptors servicing the disk units that house the journal receiver. One example of this is when you use the OptiConnect for OS/400(R) bus transport method. Having mirrored or unprotected disk units in the ASP on the target system would eliminate this effect.
4. **Processor utilization on the source system**

   The higher the processor utilization of the source system, the greater the chance of affecting the performance for the catch-up phase.
5. **Processor utilization on the target system**

   The higher the processor utilization of the target system, the greater the chance of affecting the performance for the catch-up phase.
6. **Delivery mode**

   The performance of the catch-up phase does **not** depend on the delivery mode that was specified, synchronous or asynchronous.

For additional information, see AS/400(R) Remote Journal Function for High Availability and Data Replication.

How the journal attributes affect the remote journal function

Reducing the size of the journal receivers on the source system will reduce the communications overhead of the remote journal function. Therefore, you may want to consider only journaling after images, not journaling open, close, or force entries. You may also want to consider using the various receiver size options, or using the minimized entry-specific data or fix length data values. See Methods to reduce the storage that journal receivers use for more details.

You can also refer to Remote journal attributes and Remote journals and auxiliary storage for more details.

Remote journals and auxiliary storage

Auxiliary storage will be required on both the source and target systems. The amount that is required will be about the same on both systems. Anything that is done to minimize the amount of auxiliary storage required on the source system will reduce the amount of auxiliary storage required on the target system. Additionally, the less auxiliary storage used, or smaller the journal receivers are, the less data is transmitted on the communications links. Therefore, the communications overhead will be reduced.

If the target system is not working for any extended period of time, enough auxiliary storage on the source system is needed to keep the journal receivers online. This will be required until the target system becomes available at which time the journal receivers can be replicated to the target and deleted from the source.

See Methods to reduce the storage that journal receivers use for more information about ways to reduce the auxiliary storage usage.

Journal receiver disk pool considerations

The **receiver configuration** is the disk pool the receiver resides in, and how the data for the receiver is spread across the disk arms within that disk pool. A remote journal receiver will have the same receiver configuration as its corresponding source receiver. If the source receiver is in a disk pool that is spread across multiple disk units, then the remote journal receiver will also be configured to use the same number of disk units. The remote journal receiver may be in a disk pool that has fewer disk units than the disk pool that contains the journal receiver on the source system. If this occurs, the remote journal receiver will still be configured as if it still had that same number of disk units as the source journal receiver. However, the data may physically be going to a fewer number of disk units.

Note: If the remote journal receiver is in a disk pool with fewer disk arms than the source journal receiver, then performance may be impacted. This is because the disk arms for the remote receiver will be moving considerably more than the disk arms will be moving for the source receiver. Therefore, we recommend that the number of disk arms is the same on the source and remote journal receivers disk pools.

Likewise, the journal receiver on the source system may be in a disk pool that has fewer disk units than the disk pool that contains the remote journal receiver. If this occurs, the remote journal receiver will not take advantage of all possible disk units on the target system.

Independent disk pool considerations

The following considerations apply if the remote journal receiver is on an independent disk pool:

- If the local system has the journaling environment in a basic, system disk pool, or independent disk pool, the remote journal can be in a independent disk pool. Likewise, if the local system has the journaling environment in an independent disk pool, the remote journal can be in a basic, system disk pool, or independent disk pool.

- The independent disk pool on the remote system must be varied on.
- The independent disk pool must be a library capable independent disk pool.
- The remote journal and remote journal receiver must be in the same disk pool group.

Determine the type of disk pool in which to place journal receivers has more information about journal receivers and disk pools. The Independent disk pools topic has detailed information about independent disk pools.

Remote journals and main storage

Providing greater amounts of main storage in the *BASE main storage pool on the source system might improve remote journal performance. Improvements are most likely in environments with one or more asynchronously maintained remote journals.

Providing greater amounts of main storage in the *BASE main storage pool on the target system will improve remote journal performance. This is especially true in a remote journal network with a high volume of activity. The additional storage will keep the number of page faults to a minimum, and reduce the impacts to the target system.

Set up remote journals

These topics describe the steps you would use to create and work with a remote journal network or environment. They discuss how to establish and maintain one remote journal that is associated with one local journal.

If you want to make a more complicated broadcast or cascade configuration, use the following steps for each of the remote journals in the configuration.

See the following topics to set up remote journals:

- Prepare to use remote journals
- Add remote journals

Prepare to use remote journals

Before establishing the remote journal environment, do the following steps.

1. Determine the extent of your remote journal network or environment.

   See Plan for remote journals.
2. Determine what **library redirection**, if any, you will be using for the remote journals and associated journal receivers. Library redirection is the ability to allow the remote journal and associated journal receivers to reside in different libraries on the target system from the corresponding source journal and its associated journal receivers.

   See Library redirection with remote journals.
3. Ensure that all selected libraries exist on the target systems. You will need to consider whether or not library redirection will be used when adding the remote journal.
4. Create the appropriate local journal if it does not already exist.

   See Set up journaling for more information about creating local journals.
5. Configure and activate the communications protocol you have chosen to use.

   See Supported communications protocols for remote journals for more information.

   After you have configured the communications protocol, it must be active while you are using the remote journal function. For example, if you are using the OptiConnect for OS/400(R) bus transport method, then the OptiConnect for OS/400 subsystem, QSOC, must be active. QSOC must be active for both the source system and the target system, and the appropriate controllers and devices must be varied on. If you are using a SNA communications transport, vary on the appropriate line, controller, and devices and ensure subsystem QCMN is active on both systems. If you are using TCP/IP, you must start TCP/IP by using the Start TCP/IP (STRTCP) command, including the distributed data management (DDM) servers.

See the Networking topic and OptiConnect for OS/400 for more detailed information.

6. If one does not already exist, create the appropriate relational database (RDB) directory entry that will be used to define the communications protocol for the remote journal environment.

Add remote journals

You must be aware if library redirection is in effect for the remote journal. If it is in effect, any library name processing will substitute the redirected library name for the library name that is used for the operation on the target system.

The following is the input that you must provide to add a remote journal to a source journal:

- The journal name and library on the source system to which the remote journal is being added.
- The remote journal name and library on the target system that is being added.
- A relational database directory entry, which identifies the target system and other necessary communications information.
- The type of remote journal to be added.
- Optionally, the journal or journal receiver library redirection.
- Optionally, the values for the journal message queue, text, delete receivers, and delete receiver delay attributes to be applied to any newly created remote journal.

Add a remote journal

Proceed as follows to add a remote journal:

1. In the iSeries Navigator window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database you want to use.
4. Expand **Schemas**.
5. Click the schema that contains the journal that you want to add a remote journal to.
6. Right-click the journal you want to add a remote journal to and select Properties.
7. On the **Journal Properties** dialog, click **Remote Journals**.
8. To add (associate) a remote journal to this journal, click **Add**.

Alternatively, you can use the Add Remote Journal (ADDRMTJRN) Command or the Add Remote Journal (QjoAddRemoteJournal) API to add a remote journal.

The remote journal does not have an attached journal receiver after the add remote journal processing completes. In addition, the journal state for the remote journal is set to *INACTIVE. A journal state of *INACTIVE means that the remote journal is not ready to receive any journal entries from the journal on the source system. During this time, journal entries can continue to be deposited or replicated into the journal on the source system. However, no entries are replicated to the newly added remote journal until you activate that remote journal. Refer to Activate the replication of journal entries to a remote journal for information about activating a remote journal.

Remove remote journals

You must be aware whether library redirection is in effect for the remote journal when you remove a remote journal. If it is in effect, any library name processing will substitute the redirected library name for the library name that is used for the operation on the target system.

You can also use one of the following to remove a remote journal:

- iSeries$^{(TM)}$ Navigator
- Remove Remote Journal (QjoRemoveRemoteJournal) API
- Remove Remote Journal (RMVRMTJRN) command

You must start iSeries Navigator, the QjoRemoveRemoteJournal API, or the RMVRMTJRN command on the source system for the journal on the source system identifying which remote journal to remove.

When using any of these methods, the replication of journal entries to the remote journal to be removed cannot be currently active. If the remote journal state is *ACTIVE, you must inactivate the replication of journal entries to the remote journal.

The remote journal, and any associated journal receivers, are not deleted from the target system when you remove a remote journal. Remove a remote journal does not initiate any processing on the target system. Once the remote journal is removed from the journal on the source system, you are responsible for deleting the remote journal and associated journal receivers, if desired.

You can add this remote journal back to the remote journal function definition for the journal on the source system.

Once a remote journal is removed, the journal receivers are no longer protected from deletion.

The following is the input that you must provide to remove a remote journal on a target system:

1. The journal name and library on the source system from which the remote journal is being removed.
2. The remote journal name and library on the target system that is being removed.
3. A relational database directory entry, which identifies the target system and other necessary communications information.

Disassociate a remote journal on a target system from a journal on a source system with iSeries Navigator by doing the following steps:

1. In the iSeries Navigator window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database you want to use.
4. Expand **Schemas**.
5. Click the schema that contains the journal from which you want to remove a remote journal.
6. Right-click the journal from which you want to remove a remote journal and select **Properties**.
7. On the **Journal Properties** dialog, click **Remote Journals**.
8. To remove a remote journal from this journal, click **Remove**.

Activate and inactivate remote journals

You must be aware if library redirection is in effect for the remote journal when you activate or inactive a remote journal. If it is in effect, any library name processing will substitute the redirected library name for the library name that is used for the operation on the target system.

Activating a remote journal means starting and then maintaining the replication of journal entries from a source journal to a remote journal. Activating a remote journal always occurs from the source system.

Inactivating a remote journal means ending the replication of journal entries from the source journal to the remote journal. Inactivating a remote journal can be performed from the source or target systems. However, the preferred method is to inactivate from the source system.

If this is the first time the remote journal is being activated, activating a remote journal creates one or more journal receivers on the target system. Activating the remote journal also establishes the connection between the source and remote journal so that journal entry replication can begin.

If the remote journal has previously been activated, the creation of additional journal receivers may or may not be required on the target system. This would occur prior to establishing the connection between the source and remote journal so that journal entry replication can resume.

See the following for instructions on activating and inactivating remote journals:

- Activate the replication of journal entries to a remote journal
- Relational database considerations for remote journal state
- Inactivate the replication of journal entries to a remote journal

Activate the replication of journal entries to a remote journal

In order to activate the replication of journal entries to a given remote journal, the following must be true:

- The remote journal that you wish to activate must not have a journal state of *ACTIVE. For instance, this might seem to be a reasonable request if you wanted to simply change the delivery mode from synchronous to asynchronous. However, the remote journal must be inactive before you can activate it.
- The remote journal that you wish to activate must not be actively replicating journal entries to other remote journals, as in a cascade configuration. You must inactivate the remote journals that are immediately downstream before activating the remote journal.

You need to provide the following input in order to activate the replication of journal entries to a remote journal on a target system:

- The journal name and library on the source system from which journal entries will be replicated.
- The remote journal name and library on the target system to which journal entries will be replicated.
- A relational database directory entry, which identifies the target system and other necessary communications information.
- The delivery mode to be used. Specify either synchronous or asynchronous delivery mode.
- The journal receiver from which to start journal entry replication which defines the starting point for journal entry replication.
- If an asynchronous delivery mode was specified, then the **sending task priority** may also be specified. If a value is not specified, the system selects a default priority, which is higher than what the user can specify for this value. Setting this value too large may cause a greater journal entry latency or lag.

To activate the remote journal, proceed as follows:

1. In the **iSeries$^{(TM)}$ Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database that contains the journal.
4. Expand **Schemas**.
5. Click the schema that contains the journal that has the associated remote journal that you want to activate.
6. Right-click the journal, and select **Properties**.
7. On the **Journal Properties** dialog, click **Remote Journals**.

You can also activate the replication of journal entries from a journal on a source system to a remote journal on a target system by using one of the following methods:

- The Change Journal State (QjoChangeJournalState) API
- The Change Remote Journal (CHGRMTJRN) command

Both the QjoChangeJournalState API and the CHGRMTJRN command must be issued from the source system.

The activation of the remote journal can be a long running process. This may occur if there are a large number of journal receivers and entries that initially must be caught-up in the remote journal.

Catch-up phase for remote journals has more detailed information about the catch-up phase.

Catch-up phase for remote journals: Catch-up refers to the process of replicating journal entries that existed in the journal receivers of the source journal before the remote journal was activated. The catch-up phase is the most efficient method of replicating journal entries to a remote journal. Control does not return to the requester of the activation of the remote journal until this catch-up processing has completed. You will want to consider this when deciding the starting point for journal entry replication.

Catch-up phase is initiated after the following actions occur:

- A request has been issued on the source system to activate a remote journal
- The system has determined which journal receivers and journal entries to replicate to the target system

There is a difference between the catch-up phase processing and the run-time synchronous or asynchronous processing. Catch-up processing replicates the following to the target system:

- Those journal entries that already exist in the journal on the source system
- Those journal entries that are deposited or replicated to the source journal during the catch-up processing

Run-time synchronous or asynchronous processing occurs as part of the actual deposit or replication of journal entries into the currently attached receiver on the source system. While in the catch-up phase, the journal delivery mode will be either asynchronous pending (*ASYNCPEND) or synchronous pending (*SYNCPEND), depending on the delivery mode that was specified.

The catch-up phase is the most efficient method of transporting journal entries to a remote journal in bulk.

The following is a high-level overview of the catch-up phase and related processing:

1. The starting point in the journal receiver on the source system is determined
2. If necessary, the system creates a receiver on the target system and attaches it to the remote journal.
3. The system replicates or completes replication for all of the journal entries that are contained in the receiver on the source system to the corresponding receiver on the target system.
4. If the receiver on the source system is the currently attached receiver, the system completes the catch-up processing by transitioning into synchronous or asynchronous remote journal function mode. Catch-up phase is complete, and control returns to the requester of the remote journal activation.

   The remote journal will now be maintained synchronously or asynchronously as additional journal entries are deposited, or replicated, into the attached receiver on the source system.
5. If the receiver on the source system is not the currently attached receiver for the journal on the source system, one of the following steps is performed:
   - If there is a next receiver within the source journal's chain of receivers, go back to step 2. Replicate journal entries by starting with the first entry in the next receiver.

     For more information about receiver directory chains, see Keep track of journal receiver chains
   - If there is no next receiver, (which indicates that a receiver chain break exists), the catch-up phase is complete. Processing does not transition into synchronous or asynchronous mode and the change journal state processing ends. A final escape message is sent indicating that processing has ended.

After the system transitions a given remote journal to either the synchronous or asynchronous remote journal function mode, the system continues to maintain that mode. This continues until the remote journal function is inactivated for that remote journal by using the Change Journal State (QjoChangeJournalState) API or Change Remote Journal (CHGRMTJRN) command, or a failure occurs. See Inactivate the replication of journal entries to a remote journal for more information about inactivating remote journals. See Work with remote journal error messages for more information about the possible failure conditions.

The replication of journal entries to an individual remote journal is performed independently from the replication of journal entries to any other defined remote journal. This is important if a given target system fails or if communications to a target system fails from a particular source system. If either one occurs, the remote journal function will end to those affected remote journals that reside on that target system and are maintained from the source system. All other remote journals that are being maintained from the source system will continue to function normally. For example, a source journal could have two remote journals on two different systems. In this situation, if the replication of entries from the source journal to the second remote journal ended, the replication of entries from the source journal to the first remote journal would not necessarily end. If a given remote journal has any type of failure, the system ends the remote journal function. Appropriate messages are signaled to either system or both systems involved, but the remote journal function for other remote journals would not be affected. Likewise, the communications line speed for a given asynchronously maintained remote journal will not affect the speed for another asynchronously maintained remote journal using a different physical transport.

Relational database considerations for remote journal state

Once a remote journal is activated, the remote journal function will work with the communications configuration defined by the specified relational database (RDB) entry as long as the remote journal is active. However, the information will be taken from the RDB at the point in time when the remote journal was activated. Therefore, even if the definition of the RDB entry is changed while a remote journal has a journal state of *ACTIVE, none of those changes will take effect immediately.

If the remote journal is inactivated, and then activated again, the new RDB entry definition will take effect. When you view the remote journal information, the RDB entry information that is displayed represents the state of the RDB entry information when the remote journal was last activated. See Display remote journal function information.

Inactivate the replication of journal entries to a remote journal

When you end replication of journal entries to a remote journal, it is recommended that the replication of entries be ended from the source system whenever possible, rather than from the target system. Usually, ending replication from the target system for a remote journal is only necessary when the source system has failed, and the system has not ended the remote journal function.

If you are inactivating an asynchronously maintained remote journal, you can request that the remote journal function be ended immediately or in a controlled fashion. For an immediate end, any journal entries which have already been queued for replication will not be sent to the remote journal. For a controlled end, any journal entries which have already been queued for replication will be sent to the remote journal. When all queued entries have been sent to the target system, the system sends message CPF70D3 to the journal message queue. The message indicates that the remote journal function has been ended. If you are inactivating a synchronously maintained journal, the remote journal function is ended immediately, regardless of whether an immediate or controlled end was requested. Similarly, if the remote journal is in the catch-up phase of processing, the remote journal function is ended immediately. This is also regardless of whether an immediate or controlled end was requested.

To inactivate the replication of journal entries proceed as follows:

1. In the **iSeries$^{(TM)}$ Navigator** window, expand the system you want to use.
2. Expand **Databases**.
3. Expand the database that contains the journal.
4. Expand **Schemas**.

5. Click the schema that contains the journal that has the associated remote journal that you want to activate.
6. Right-click the journal, and select **Properties**.
7. On the **Journal Properties** dialog, click **Remote Journals**.
8. On the **Remote Journals** for dialog, select the remote journal in the list of remote journals, and then click **Deactivate** to inactivate the selected remote journal.

You can also use the Change Journal State (QjoChangeJournalState) API and Change Remote Journal (CHGRMTJRN) command to inactivate the replication of journal entries to a remote journal. For this purpose, the API can be initiated from either the source system or the target system. The CHGRMTJRN command can only be initiated from the source system. You can also use the Change Journal (CHGJRN) command on the target system to inactivate the remote journal.

Manage remote journals

Managing the remote journal function requires basic tasks such as:
- Keeping records of your remote journal network.
- Evaluating the impact on the remote journal network as new applications are added or the system workload grows.
- Considering the ramifications of journal receivers on two systems which require regular save and delete processing.
- Considering the save and restore implications of the remote journal network.

The following information describes the management tasks for remote journals:
- Keep records of your remote journal network
- Display remote journal function information
- Evaluate how system changes affect your remote journal network
- Get information about remote journal entries
- Journal receiver management with remote journals
- Swap journal receiver operations with remote journals
- Considerations for save and restore operations with remote journals
- Remote journal considerations when restarting the server
- Work with remote journal error messages

Keep records of your remote journal network

Always have a current list of the remote journals that are associated with local journals, and their associated communications information.

For each journal which has remote journals associated with it, use the following command: `WRKJRNA JRN(`*`library-name/journal-name`*`) OUTPUT(*PRINT)`.

Alternatively, you can use the Retrieve Journal Information (QjoRetrieveJournalInformation) API to retrieve the information and place it in a file.

To get the related relational database information, use the following command: `WRKRDBDIRE RDB(*ALL) OUTPUT(*PRINT)`.

Remember to do this for all cascaded remote journals as well, not just the local (or primary) system.

Display remote journal function information

When you are working with the remote journal function, you will want to be able to view the remote journal network. You may also want to view the various attributes, journal states, or delivery modes. The Work with Journal Attributes (WRKJRNA) display includes the list of all remote journals that are associated with a given journal. When looking at a specific journal, you can see information about the journal’s source journal, if any. Additionally, you can see all remote journals which are immediately downstream from the specified journal. If those remote journals are cascaded to other remote journals, you will not be able to see any cascaded remote journal information. To see that information, you must invoke the WRKJRNA command for that remote journal on its own system. This information is also available through the Retrieve Journal Information (QjoRetrieveJournalInformation) API.

Additionally, the Display Journal Receiver Attributes (DSPJRNRCVA) displays provide additional information about the remote journal characteristics of the journal receivers. The DSPJRNRCVA command also has an API counterpart to allow program retrieval of the journal receiver information, the Retrieve Journal Receiver Information (QjoRtvJrnReceiverInformation) API.

Evaluate how system changes affect your remote journal network

After you have initially established your remote journal network, you need to keep up with changes that occur on the system.

If the amount of work that is going to the journals which you are replicating increases, you may need to consider upgrading the communications method.

The traffic rate for work other than the remote journal function may increase on a communications method that is shared. If this occurs, you may need to consider separating the various pieces of communications traffic so that the remote journal function is not impaired. This is especially important if you are using the synchronous delivery mode.

An application that is being protected may become more critical to your business, where any time that the system is not working is considered disastrous. If this occurs, you may need to consider upgrading that application’s remote journal to using the synchronous delivery mode so that no journal entries are lost.

Get information about remote journal entries

Working with the journal entries in a remote journal is essentially the same as working with the journal entries in a local journal.

The exceptions are explained in the following topics:

- File identifier considerations for working with integrated file system entries
- Confirmed and unconfirmed journal entries
- Journal entries from a remote journal with library redirection
- Retrieve journal entries from a remote journal during the catch-up phase
- Remote journal considerations for retrieving journal entries when using commitment control
- Remote journal considerations for retrieving journal entries when using journal caching

In addition, the system name, date, and time stamp in the journal entries are based on the original local journal. They are not based on the system of the remote journal where the entries are viewed.

See Display information for journaled objects, journals, and receivers and Work with journal entry information for more information.

File identifier considerations for working with integrated file system entries: » If you plan to replay the integrated file system operations in the remote journal to objects on the target system, and if you primed that target system with objects that were restored from the source system, then some additional considerations apply to replaying those journal entries. Integrated file system journal entries on remote journals are only identified by the file identifier in the object name field. They are not identified by path name. When you restore an integrated file system object on a remote system, the remote system does not maintain the same file identifier that was used on the source system. It assigns that object a new file ID. However, the journal entries in the remote journal receiver refer to that object’s original file ID. Therefore when you replay the journal entries you cannot use the file ID on the remote journal to find the path of the object. That file ID will either not exist or be the file ID for the wrong object.

To prevent potential problems, it is recommended that you create a table that maps the old and new file IDs with the object's path. The map can be something like the following table:

Object path	Source file ID	Target file ID
/myFolder/subFolder/MyObject	123456...	789123...
/myNextFolder/anotherFolder/MyObject2	654321...	321987...

Collect the information for mapping file IDs

You can use different methods to determine the file IDs:
- Use local journaling on the target system where you restore the object.
- Use the object's path to find its file ID with the Get Attributes (Qp0lGetAttr()) API on the source system.
- Use the object's file ID to find its path with the Get Path Name of Object from Its File ID (Qp0lGetPathFromFileID()) API on the source system.

Use local journaling on the target system

If an object is journaled when you restore it to the target system, a B FR journal entry is deposited on the target system's local journal receiver. The entry-specific data of the B FR journal entry contains the following:
- Media file identifier—the file ID of the object on the media. This file ID is the same as the object's file ID on the source system.
- Restored file identifier—the object's new file ID after it is restored to the target system.
- Restored over file identifier—the file ID of the object that was restored over.

If you are concerned about the demand that journaling puts on the remote system's resources and storage space, you can put the journal in *STANDBY state. Even though the journal is in standby state, the system still deposits B FR entries.

Use the object's path to find its file ID with the Qp0lGetAttr() API

On the source side, if you know the object's path but do not know its file ID, you can use the Qp0lGetAttr() API to get the file ID. This is especially helpful if you do not want to use journaling on the remote system. You then need to send that information over to the target system to update the table which must exist on the target system.

Use the object's file ID to find its path with the Qp0lGetPathFromFileID() API

On the source side, if you know the object's file ID, but do not know it's path, you can find it using the Qp0lGetPathFromFileID() API. You can then use this path to replay the journal entries on the target system, assuming that the path on the target system is the same as the path on the source system. This API will only return an absolute path name of the object. If the object has more than one path name, the API only returns one path. You then need to send that information over to the target system to build the table which must exist on the target system.

Maintain the table as the replicator job applies journal entries

Once you have the table created, you must keep it updated. One way to keep the table updated is to update the table as the replicator job applies journal entries. On the target system, when the replicator job applies entries to do operations such as creating objects, adding links, or removing links, the journal entry information in these entries has the path name and file ID in it at that time. As the operation is replayed you can use this information to build the table on the target system. «

Confirmed and unconfirmed journal entries: For a local journal, all entries are confirmed entries. There is no concept of unconfirmed entries.

For a remote journal that is maintained asynchronously, all entries are confirmed entries. For a remote journal that is maintained synchronously, there are both confirmed and unconfirmed entries. Unconfirmed entries will only become important if you are using the remote journal support for a hot-backup or data replication environment, and the source system has a failure such that the target system will take over processing.

Confirmed journal entries are journal entries replicated to a target system, and the state of the I/O to auxiliary storage for the same journal entries on the primary system is known to have completed.

Unconfirmed journal entries are entries replicated to a target system, but the state of the I/O to auxiliary storage for the same journal entries on the primary system is not known. Unconfirmed entries only pertain to remote journals that are maintained synchronously. The remote I/O to the remote journal is overlapped with the local I/O to the local journal for better performance. Such journal entries on the target system are held in the data portion of the journal receiver. However, the journal entries are not officially included with the remainder of the journal entries until the confirmation of the I/O for the same entries is received from the primary system. For performance reasons, confirmation of these entries is not typically sent to the target system until some later delivery of journal data to the target system.

While the journal entries are unconfirmed on a target system, the entries typically cannot be retrieved from the remote journal. You can retrieve the journal entries by using the INCENT(*ALL) parameter on the following commands:

- Display Journal (DSPJRN)
- Retrieve Journal Entry (RTVJRNE)
- Receive Journal Entry (RCVJRNE)

You can also retrieve the journal entries by specifying *ALL for the include entries key for the Retrieve Journal Entries (QjoRetrieveJournalEntries) API. The INCENT(*ALL) parameter, or include entries key specification of *ALL, requests that all confirmed and unconfirmed entries are included. This means that for synchronous remote journal function, the last few journal entries are not immediately retrievable from the remote journal by using the default command invocations. This is true even though all journal entries physically reside in both the local journal and the remote journal. This is done so that application programs do not make decisions on the target system by using journal entries that may not end up being deposited into the local journal. This is because those journal entries would not cause a change to the original data.

With respect to a hot-backup application apply, in most circumstances only the confirmed journal entries in the remote journal are of interest. In the data replication environment, a hot-backup application apply would probably never want to apply any unconfirmed journal changes. This is because any subsequent activation of the remote journal will ensure that the journal entries in the remote journal will match the journal entries in the source journal. However, as described in Scenario: Recovery for remote journaling, knowledge of the unconfirmed journal entries is essential during the switch-over and switch-back processing for a hot-backup environment.

When a remote journal is inactivated, all unconfirmed entries are removed from the remote journal. It is important that those entries are retrieved prior to the remote journal being inactivated, if those entries are desired for additional processing on the backup system. The message that is sent to the journal message queue when the remote journal is inactivated by the system will indicate if the remote journal has any unconfirmed journal entries. See Work with remote journal error messages.

Journal entries from a remote journal with library redirection: All journal entries that are retrieved from a remote journal will have the object names as they exist on the local system.

The following journal entries will show the name of the journal receiver as it was on the local system even if the entry is displayed on a remote system. This is because these entries really apply to the version of the journal receiver that existed on the local system.

- J PR - Previous Receiver entry
- J NR - Next Receiver entry
- J RD - Receiver Deleted
- J RR - Receiver Restored
- J RS - Receiver Saved
- J RF - Receiver Saved with storage Freed
- Object saved entries - See the Journal entry information finder for a list of the possible entry types.
- Journal changes applied entries - See the Journal code finder for a list of the possible entry types.
- Journal changes removed entries - See the Journal code finder for a list of the possible entry types.

Retrieve journal entries from a remote journal during the catch-up phase: During the catch-up phase, journal entries that have been replicated to the target system can be retrieved from the remote journal.

You can activate and inactivate the remote journal function while concurrently running the following commands to view journal entries on the target system:

- Display Journal (DSPJRN)
- Retrieve Journal Entry (RTVJRNE)
- Receive Journal Entry (RCVJRNE)
- Retrieve Journal Entries (QjoRetrieveJournalEntries) API

When the remote journal is in the process of being caught-up from the attached journal receiver on the source system, two things can happen with respect to objects and their names in the journal entries.

- If journaling is started for any objects on the source system, the object name that is given on the target system in the start journal entry may be *UNKNOWN.
- If any move or rename operations take place, the last object name that was known before the catch-up phase started is what will be given. The actual new name may not be available until the catch-up phase is complete.

If you are using the DSPJRN or RTVJRNE command, additional informational messages will indicate that this situation occurred. If you are using the RCVJRNE command, additional information is provided on the exit program interface to help distinguish these situations as well. If you are using the QjoRetrieveJournalEntries API, additional information is provided in the returned data to help distinguish these situations. When necessary, the system attempts to minimize the possibility of showing these inconsistencies by temporarily delaying the processing performed by these commands.

Once the catch-up phase is completed, these inconsistencies will be resolved, and complete information will again be available.

Remote journal considerations for retrieving journal entries when using commitment control: Special performance related processing is done by the system when depositing entries that are associated with commitment control transactions to a local journal. When a job deposits a journal entry that is not associated with a commitment control transaction, that job waits for the local journal I/O to auxiliary storage to complete. After completion, control is given back to the application. A different technique is used for those journal entries that are associated with a commitment control transaction which results in the application being given control back before the local journal I/O is complete. This special processing has some ramifications when you retrieve journal entries from a remote journal.

For journal entries deposited related to a commitment control transaction, a job only waits for the local journal I/O to complete when the following journal entries are being deposited into the local journal:

- Journal code C, journal entry type CM (Commit)
- Journal code C, journal entry type RB (Rollback)

For remote journals, those journal entries that the job that is making the deposit does not wait for are not immediately replicated or scheduled to be replicated to the remote journal. Prior to the CM (Commit) or RB (Rollback) entry being deposited, there is no guarantee as to when the journal entries for open commitment control transactions will be retrievable from the remote journal.

After the commit or rollback operation is complete for a particular commitment control transaction, all journal entries associated with that transaction are immediately retrievable from an asynchronously maintained remote journal. However, there may be some journal entry delivery latency due to the transport method that is being used.

For a synchronously maintained remote journal, all journal entries associated with the commitment control transaction are assured to be retrievable after the CM (Commit) or RB (Rollback) entry is deposited.

Interspersed local journal I/O, for journal entries not associated with a commitment control transaction, can also affect when the journal entries associated with a commitment control transaction can be retrieved from the remote journal. In this I/O a job actually waits for the local journal I/O to complete. This interspersed local journal I/O will also cause the journal entries related to the commitment control transaction to be replicated to the remote journal. Once in the remote journal, and when later remote journal I/O makes them confirmed, the journal entries that are related to the commitment control transaction are retrievable.

Note: These considerations also apply if you generated entries that use the Send Journal Entry (SNDJRNE) command or Send Journal Entry (QJOSJRNE)API. If the application or user never requests to force these user generated entries, they will only be replicated to the remote journal when some other action forces the journal entries. Therefore, you will wish to periodically specify FORCE(*YES) when using these send journal entry functions.

These considerations also apply to any database physical file open or close journal entries; or directory or stream file open, close, or force entries.

Remote journal considerations for retrieving journal entries when using journal caching: When you use journal caching for the local journal, the system performs special performance related processing when it deposits journal entries. With journal caching, the system waits longer to write journal entries to disk, leading to fewer but larger disk writes. This action helps performance, but also delays the journal entries from being sent to the target system, even if you are using synchronous remote journaling.

Journal receiver management with remote journals

As with local journals, regularly save and delete your journal receivers to minimize the amount of online auxiliary storage which is used by the journal receivers. The change journal processing for a remote journal is driven by the source journals change journals. See Swap journal receiver operations with remote journals for more information.

If you plan to move the responsibility for storing journal receiver data from the primary system to the remote system, you can elect to quickly delete journal receivers from the primary system after they have been replicated to the backup system with automatic deletion of journal receivers. On your backup system, you can then select to not use the automatic deletion of journal receivers on the remote journal, and manage the receiver save processing as you did before. Remember that once you add a remote journal, you cannot delete the source journal receiver until it has been replicated to all associated remote journals. Any journal receivers that are attached subsequently are also protected. The protection is eliminated when you remove the remote journal. If you have cascaded remote journals, consider using automatic deletion of journal receivers on the local journal, and on the lowest level remote journal. You would then not use automatic deletion of journal receivers on the cascaded remote journal since you plan to do your save processing on that system.

The Delete Journal Receiver exit point, QIBM_QJO_DLT_JRNRCV can be of assistance as well. For example, you might want to add an exit program to QIBM_QJO_DLT_JRNRCV which verifies that the journal receiver is no longer needed for any hot-backup application apply processing before it can be deleted. Refer to Delete journal receivers for information about this exit program.

Swap journal receiver operations with remote journals

To swap journal receiver on a remote journal, perform a swap journal receiver operation on the source system to attach a new receiver to a local journal. When this happens, the remote journal function automatically attaches a new receiver to those remote journals that are currently being maintained synchronously or asynchronously. If the journal sequence numbers were reset as part of the swap journal receiver operation performed for the local journal, the remote journal function will also reset the journal sequence number for each remote journal. This keeps the journal sequence numbers synchronized between the local journal and the remote journal. For remote journals that are being synchronously maintained, a coordinated swap journal receiver operation is performed for the local journal on the source system and the remote journals on the target systems. For asynchronously maintained remote journals, the new receiver is attached when the target system receives the journal entry with journal code 'J' and entry type 'PR' (previous receiver).

If the swap journal receiver operation fails on the target system, the remote journal function ends for that remote journal, and processing continues on the source system. The system sends a message to the journal message queue that indicates that the remote journal function failed. When applicable, the system sends remote journal failure type messages to the related journal message queues on both the affected source and target systems. See Work with remote journal error messages for more information.

You cannot initiate a swap journal receiver operation to attach a new receiver directly for a remote journal. New journal receivers are always attached to the remote journal by the remote journal function as new receivers are attached to the local journal. However, you can perform a change journal operation on a remote journal to change several other attributes for the remote journal such as the journal message queue or delete receivers value.

A swap journal receiver operation to attach a new receiver to a local journal that has an associated remote journal in the catch-up phase can be performed. This is regardless of whether the remote journal is currently being caught-up from a detached or the currently attached receiver on the local system. The catch-up phase of processing will not transition into synchronous or asynchronous delivery mode until the end of the currently attached receiver for the local journal is reached.

Considerations for save and restore operations with remote journals

The following information describes general considerations for save and restore operations with remote journals:

- Rules for saving and restoring journals
- Rules for saving and restoring journal receivers
- File identifier considerations for working with integrated file system entries
- Considerations for restoring journaled objects
- Considerations for restoring objects saved with SAVSTG

Rules for saving and restoring journals: It is recommended that you save the remote journal network after the addition of any and all remote journals that will be associated with the journal. This includes saving the local journal and any associated remote journals, as well as the journal receivers that are associated with the local journal.

Follow the basic save and restore rules for journals that are listed here:

- A saved local journal is always restored as a local journal.
- A saved remote journal is always restored as a remote journal.
- As with all prior save and restore support for journals, the support will not allow a restore-over operation for a journal. This is true for both local and remote journals.
- When restored, a local or remote journal is always restored to the library from which it was saved. For a local journal, this library is referred to as the original journal library. For a remote journal, this library is referred to as the redirected journal library.

  For a remote journal, library redirection may not have been specified when adding the remote journal to the local journal's definition. If this occurs, then the redirected journal library name is the same name as the original journal library name.

Note: This is always true except in the case where the journal was saved from library QRCL. (The journal could reside in library QRCL due to prior Reclaim Storage processing.) In that case, the RSTLIB parameter must be specified on the restore request, and you must specify the library where the journal originally resided. For a local journal, this is existing support and is not new. For a local journal, the library that must be explicitly specified is the original library.

  This support logically extends to remote journals. For a remote journal, the redirected library must be explicitly specified on the RSTLIB parameter of the restore request.

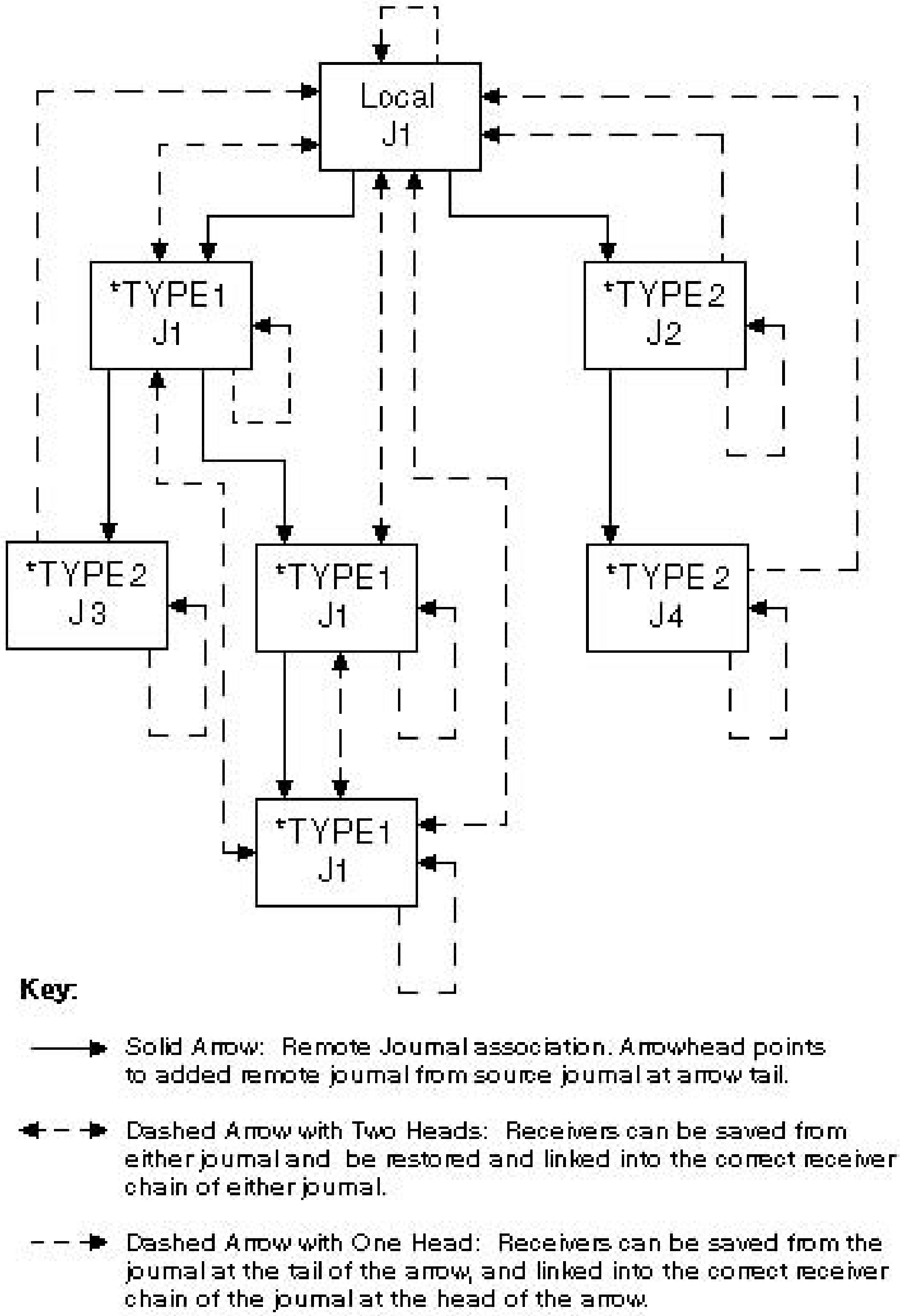
- If remote journals are associated with a journal when a journal is saved, the information that is related to the added remote journals is also saved.

  When the journal is restored, the information that is saved about its remote journals is also restored. This information is included as part of that journal's definition. This is true whether the journal being saved is a local or a remote journal. When restored, the restored journal's definition will only include the saved, immediately downstream remote journal definitions.

Note: None of the actual downstream remote journals are actually verified as part of the restore operation. Any necessary validation of the remote journal information occurs when you activate that particular remote journal by using the Change Journal State (QjoChangeJournalState) API or Change Remote Journal (CHGRMTJRN) command.

- Local journals are restored to the same state in which they are saved.

Rules for saving and restoring journal receivers: The following figure illustrates the restore relationships for journal receivers that are associated with remote journals, based on the remote journal type.



Key:

Solid Arrow: Remote Journal association. Arrowhead points to added remote journal from source journal at arrow tail.

Dashed Arrow with Two Heads: Receivers can be saved from either journal and be restored and linked into the correct receiver chain of either journal.

Dashed Arrow with One Head: Receivers can be saved from the journal at the tail of the arrow, and linked into the correct receiver chain of the journal at the head of the arrow.

There are several unique rules which govern where the journal receivers that are associated with a remote journal can be restored. The rules also discuss the placement of the journal receivers in the receiver directory chain of a local or remote journal. These rules are influenced by the remote journal type of the journal to which the journal receiver was attached. These rules are also influenced by the library redirection that was in effect when that receiver was attached. See Types of remote journals.

Note: You can always save receivers from a journal, and then restore the receivers to another local journal of the same name. However, they will be placed in their own separate receiver chain.

The following items describe the rules that the system uses when restoring journal receivers:

1. The system first attempts to find an appropriate remote journal. When searching for a remote journal, the system follows the following rules:
   a. If the saved receiver was originally associated with a local or *TYPE1 remote journal, then the system searches for a *TYPE1 remote journal.
      1) If a *TYPE1 remote journal was defined at the time this receiver was attached, then use the journal and receiver library redirection that was in effect and saved with the receiver. If no *TYPE1 remote journal was defined at the time this receiver was attached, then the original journal library and receiver library names will be used when searching for the *TYPE1 remote journal.
      2) If a *TYPE1 remote journal is found, and the current receiver library redirection for the found *TYPE1 remote journal matches the library name where the receiver is being restored, the journal receiver will be associated with the found *TYPE1 remote journal.
   b. If the receiver was originally associated with a *TYPE2 remote journal, then the system searches for a *TYPE2 remote journal. When searching for the *TYPE2 remote journal, a journal with the same name as the name that was saved with the receiver will be used.
      1) The journal receiver will be associated with a found *TYPE2 remote journal if the following conditions are met:
         - A *TYPE2 remote journal is found with the correct name in the correct library.
         - The found journal is in the exact same remote journal network as that of the saved receiver.
         - The receiver is being restored to the same named system or same named ASP group as the name of the system or ASP group at the time the receiver was saved.
2. If a remote journal was not found, then the system searches for a local journal. When searching for a local journal, the original journal and journal library names are used.

   The journal receiver will be associated with a found local journal if the following conditions are met:
   - A local journal is found by the correct name in the correct library
   - The original journal receiver library name for the found journal matches the library name where the receiver is being restored
3. If a local journal cannot be found, the restore operation will be allowed to proceed. The journal receiver will not be associated with any journal, if the receiver is being restored to the original or redirected receiver library.
4. Still honoring the previous receiver restore rules, the following must also be true if the receiver is being restored over an existing receiver:
   a. If the receiver is not being associated with any journal (as previously determined from the prior receiver restored rules), then following items apply:
      1) The receiver creation time stamps must match.
      2) If the saved receiver was ever associated with a journal, then it must have been previously associated with a journal of the same type as that of the existing receiver.
      3) If the saved receiver was ever associated with a remote journal network, then it must have been previously associated with the same remote journal network as that of the existing receiver.
      4) The saved receiver must have at least as many entries as the existing receiver.
   b. If the receiver is being associated with a local journal, then the following items apply:
      1) If the saved receiver was originally associated with a local journal, then the receiver creation time stamps must match.
      2) If the saved receiver was not originally associated with a local journal, then the saved receiver must have been originally associated with the same remote journal network as that of the existing receiver.
      3) The saved receiver must have at least as many entries as the existing receiver.
   c. If the receiver is being associated with a *TYPE1 remote journal, then the following items apply:

    1) The receiver creation time stamps must match, and the saved receiver must have been originally associated with a local or *TYPE1 remote journal.

d. If the receiver is being associated with a *TYPE2 remote journal, then the following items apply:

    1) The receiver creation time stamps must match, and the saved receiver must have been originally associated with the same *TYPE2 remote journal.

When receivers are saved from or restored to a target system and associated with a remote journal, no journal entries are deposited to indicate that the save or restore occurred. However, the object save and restored date and time stamps are updated accordingly.

Save and restore considerations

»

Considerations for remote journal receivers Do not save the receiver while it is attached to the remote journal. If it is a long running save it can inhibit a change journal operation that was initiated by the source and the remote journaling environment can time out and fail. «

Nonreplicated journal receiver protection considerations
The protection provided, which prevents journal receivers that are not fully replicated to all associated remote journals from being deleted, is removed when the journal receiver is restored.

Unconfirmed journal entries save considerations
When a journal receiver that is associated with a remote journal is saved, only those journal entries which have been confirmed are saved to the media. Therefore, no unconfirmed journal entries, nor any journal entries that would not survive any IPL journal recovery processing, will be saved.

Journal receivers saved with STG(*FREE) considerations
Even if a journal receiver has not been fully sent to all known remote journals, such a journal receiver can be saved with STG(*FREE). However, a diagnostic message is left in the job log indicating the freeing of the journal receiver storage without the journal receiver first being fully replicated to all downstream remote journals. This is in contrast to the default action taken when attempting to delete a receiver that has not been fully replicated to all downstream remote journals.

Considerations for restoring journaled objects: » For an object that is restored and associated with a local journal in standby state, journaling starts for that object, but no restore entry is deposited in the journal receiver. If the object is being restored-over and is currently journaled to a local journal in standby state, the restore is not prevented, and no restore entry is deposited in the journal receiver. «

The system will send a diagnostic message for any object in which the 'object restored' journal entry cannot be sent due to a problem with the journal or attached journal receiver, unless the journal is in standby state. The system always attempts to start journaling for an object that was journaled at save time to the same named journal, in the same named library, during a restore operation. This is still true, and there are no processing changes to note if a local journal is found by the restore processing. However, if a remote journal is found by the restore processing, the restore is completed successfully, but journaling is not started for the restored object. A diagnostic message is sent that indicates that a remote journal was found by the restore processing. This message is followed by the message that is already sent that indicates journaling was not started.

In a hot-backup configuration, a local journal is used on the backup system to capture the changes that are made to the objects on the remote system. This occurs when the remote system is logically promoted to assume the role of the primary system. The local journal that is being used on a backup system might not be in the exact same-named library as the journal that is being used for the object at save time. If this occurs, you are responsible for starting journaling for the restored objects. This is a fundamental reason to use library redirection for all defined remote journals.

Considerations for restoring objects saved with SAVSTG: If you restore a system from Save Storage (SAVSTG) media, the primary remote journal function concerns have to do with configuration changes involving additionally defined remote journals. These remote journals were established after the SAVSTG media was produced. If a primary system is restored from SAVSTG media, journal receivers can be restored back to the primary system from versions saved from any of the associated remote journals in the remote journal environment. If a backup system is restored from SAVSTG media, then the catch-up phase for activating the remote journal can replicate all necessary journal receivers that are still online from the primary system to the restored backup system. Those journal receivers that are not online, and were attached to a *TYPE1 remote journal, can be restored back to the backup system. They can be restored from any saved versions of the journal receivers that were previously taken from one of the following:

- The primary system
- Any of the associated remote journals in the remote journal environment

See Rules for saving and restoring journal receivers for the journal receiver restore rules which will be used for such a restore.

Another consideration occurs as part of the processing that is performed by the system when restoring journal receivers. Before associating a journal receiver with a local journal and retaining any remote journal information, the journal library name, and the system name or the independent disk pool name must be correct. This allows the system to differentiate between a local journal that was originally created and one that was restored to a different physical system using SAVSTG media. This case assumes that the user assigns a new system name as part of the SAVSTG procedure.

A related case can potentially cause trouble. In this case, the system was also restored using SAVSTG media. However, the system it was restored to was not the same physical system but it still had the same name as the system from where the media was produced. Avoid duplicating this situation.

Remote journal considerations when restarting the server

The following are considerations for remote journaling when you restart the server.

Considerations for restarting replication of journal entries

The replication of journal entries to each of the associated remote journals ends implicitly when the local system ends. To begin replicating journal entries the remote journal, you must inactivate the remote journal on the target system, then activate it again. After an IPL or vary on operation, you are not required to reassociate the desired remote journals with the journal on the source system.

Considerations for main storage preservation

In addition to unconfirmed I/O for journal entries, you also need to consider the preservation of main storage for a failed system during recovery processing. Given certain system failures, main storage might or might not be preserved during the following IPL to recover from the system failure. Therefore, it is possible for journal entries to survive in a local journal after a system failure, even if the local or remote I/O was never performed for those journal entries.

Therefore, IPL recovery on a primary system might preserve changes that are not yet replicated to any of the remote journals, even the remote journals that are synchronously maintained. Scenario: Recovery for remote journaling demonstrates that you can use the remote journal function to account for journal entries that survive a system failure in this manner. These journal entries do not cause a total re-priming of the original data when switching back from a backup system which took over the role of the primary system.

In the scenario, when the system ends, the system does not return control to the application programs that are in the process of generating these surviving journal entries. Therefore, the application does not know whether or not any of operations completed when the system ends. Also, the application does not make dependencies or decisions on these operations. This includes dependencies or decisions by the application performing the operation or any other application that could be possibly dependent upon the data affected by the operation.

Because of this consideration, it is recommended that you journal both the before-images and after-images for any objects, if possible. With the before-images, the work can then be backed out after the IPL or vary on operation. If the data activity is not backed out after the IPL or vary on operation, the alternative is to re-prime the primary system data completely from the backup data which had assumed the role of the primary.

Considerations for when the target system ends

When remote journaling is active, neither a normal end nor an abnormal end of the target system affects journaling on the source system. The local system continues to deposit entries into the local journal without an error. The system sends a message to the local journal's message queue to alert the operator that remote journaling ended. When the target is again available, you can reactivate remote journaling from the source system. When you activate remote journaling, the default is for the local system to start sending journal entries starting with the first entry the target system is missing.

Considerations for commitment control

Commitment control, especially two-phase commitment control, can cause some additional considerations and potential complications. For example, if any of the entries that were preserved but not yet confirmed were a commit or a rollback operation, then the transaction will have to be reconciled accordingly between the primary system, and the backup system.

Considerations for journal caching

Journal caching affects remote journaling. Since journal entries are not sent to the target system right away, the number of journal entries that are not confirmed are always greater than if you are not using journal caching.

Work with remote journal error messages

Several different error conditions can occur when the remote journal function is active. When an error condition is encountered, the system automatically ends the remote journal function on the source system to that remote journal. The system notifies you that a failure occurred. Failure notification is made on both the source system and the target system. Notification is made by sending a message to the journal message queues associated with the source and target journals as appropriate.

In some cases where the source system fails, you may have to inactivate the remote journal on the target system. (You inactivate the remote journal on the target system by using the Change Journal State (QjoChangeJournalState) API or the Change Journal (CHGJRN) command.) For a synchronously maintained remote journal, inactivating the remote journal on the target system will discard all unconfirmed journal entries. See Confirmed and unconfirmed journal entries for more information.

Note: The source system might not detect that an error has occurred on the target system until the next time a journal entry is replicated to that failing target system.

Additional messages can be sent to the journal message queue for normal remote journal processing. For example, if you requested a controlled inactivate of the remote journal, a message will be sent to the message queue when the inactivate processing has completed.

Even though the remote journal function has been ended, the local journal is not automatically inactivated. Therefore the local system journal entry deposits will continue normally.

The remote journal function messages that are sent to the journal message queue are listed as follows:

CPF70D3
A controlled inactivate of a remote journal has completed.

CPF70D4
The remote journal function is no longer active due to various reasons. For a synchronously maintained remote journal, there may be unconfirmed entries which may need to be processed prior to the remote journal being inactivated.

CPF70D5
The remote journal function is no longer active and has been ended due to various reasons. There are no unconfirmed entries.

CPF70D6
The remote journal function was ended due to storage constraints.

CPF70D7
There was a problem on the target system while attempting to execute a change journal.

CPF70DB
A severe error has occurred with the remote journal function, and service must be notified.

CPF70DC
There was a timeout on the target system while attempting to attach a new journal receiver to the remote journal.″

Display the messages on your system for more information.

Scenarios: Remote journal management and recovery

These scenarios describe the possible ways that JKL Toy Company can use remote journal management. JKL Toy Company uses the server JKLINT as their web server.

They need 24x7 availability for the critical data on this server, and they accomplish that by having a second server, JKLINT2, that shadows JKLINT. They use a high availability replication solution to copy the data from JKLINT to JKLINT2. Then, if JKLINT goes down, they can switch to JKLINT2.

The following scenarios describe two possible environments in which they can use remote journaling. The first scenario describes how JKL Toy Company can set up a data replication environment. The second scenario describes how they set up a hot-backup environment. The third scenario describes recovery steps if one of the servers fails.

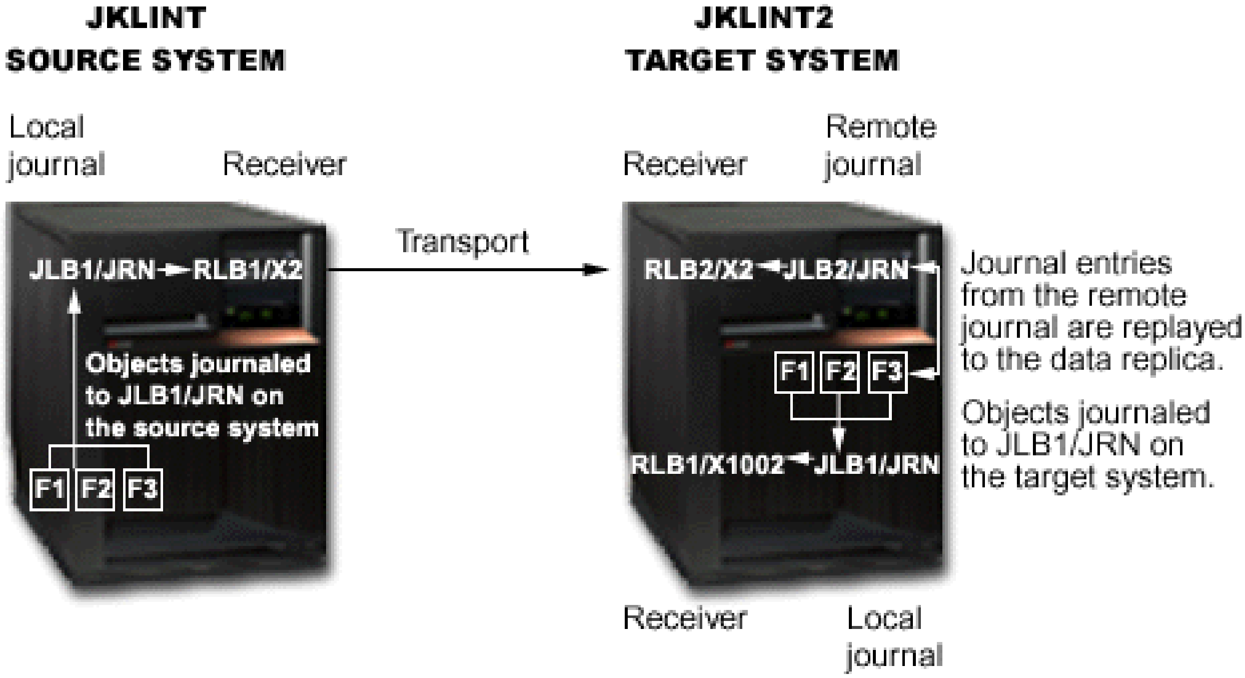
- Scenario: Data replication environment for remote journals
- Scenario: Hot-backup environment
- Scenario: Recovery for remote journaling
- Details: Recovery for remote journaling scenario

For a complete description of JKL Toy Company’s network, see Scenario: Journal management.

Scenario: Data replication environment for remote journals

In this scenario, JKLINT and JKLINT2 use remote journaling for data replication purposes only. The following figure illustrates this remote journaling environment. **Data replication** is the function of maintaining a separate copy of data from an original copy, keeping the two copies consistent with each other.

Typical data replication environment with remote journal function


JKLINT
SOURCE SYSTEM
Local journal
Receiver
JLB1/JRN
RLB1/X2
Objects journaled to JLB1/JRN on the source system
F1
F2
F3
Transport
JKLINT2
TARGET SYSTEM
Receiver
Remote journal
RLB2/X2
JLB2/JRN
Journal entries from the remote journal are replayed to the data replica.
F1
F2
F3
Objects journaled to JLB1/JRN on the target system.
RLB1/X1002
JLB1/JRN
Receiver
Local journal

How the data replication environment works

Local objects, F1, F2, and F3, on JKLINT are journaled to local journal JRN in library JLB1. A remote journal is defined on JKLINT2, where JRN has been redirected to library JLB2. This remote journal receives journal entries from the local journal on JKLINT. A hot-backup application apply replays the changes to the data replica on system JKLINT2.

The data replica is journaled to a local journal, JRN in library JLB1, for system recovery purposes only, so this journal must be in active state. If system JKLINT2 fails, the system performs recovery for the objects by using this local journal.

A hot-backup application assists in replicating data from one system to another. The hot-backup application apply is only performing the replay of operations to the data replica on the target system.

Since this scenario is for a data replication environment, the hot-backup application does not perform a switch-over to the backup system. See Scenario: Hot-backup environment for more details about hot-backup applications applies and hot-backup switch-overs.

How to establish the data replication environment for JKLINT and JKLINT2

The objects and local journal on JKLINT are already assumed to exist. The journal state for the local journal is also assumed to be active. The communications environment and associated RDB entries already exist and are established.

Establishing the data replication environment for JKLINT and JKLINT2 requires the following:

1. Create the remote journal on JKLINT2, and specify library redirection. Library redirection indicates that the journal's library, JLB1 on JKLINT, is redirected to library JLB2 on JKLINT2. The journal receiver's library, RLB1 on JKLINT, is redirected to library RLB2 on JKLINT2.

   After this step, the remote journal exists, but no receiver is currently attached.

2. To establish a clean breakpoint, perform a change journal operation to attach a new journal receiver at this time.
   Notes:
   a. The next step restores local journal JRN in library JLB1 and attaches receiver X1002 in library RLB1. It then restores the objects, and starts journaling for the objects to the restored local journal.
   b. Since it is not possible to save and restore the contents of data queues, you must take that into consideration when priming the data replica for any data queue objects.
3. Save the local journal and objects from JKLINT and restore them to JKLINT2. This primes the data replica and establishes the local journaling environment on JKLINT2.
4. Activate the remote journal on system JKLINT2. Specify that the remote journal must start with the attached receiver. Since no receiver is attached to the remote journal, the receiver that is currently attached to the local journal on JKLINT (X2) is created on JKLINT2. This receiver is then attached to the remote journal. Journal entries are replicated, starting with the first journal entry in receiver X2.

   An additional parameter on the Change Journal State (QjoChangeJournalState) API and Change Remote Journal (CHGRMTJRN) command indicates whether the remote journal function is to be maintained synchronously or asynchronously. Depending on how the remote journal is maintained, other parameters may also apply.
5. The hot-backup application apply process receives or retrieves journal entries from the remote journal, starting with the entries that were deposited after the data was saved and primed into the data replica. The process then starts replaying the changes that are contained in these journal entries to the data replica.

Normal run-time environment for the data replication environment

You can activate and inactivate the replication of journal entries to the remote journal as needed. Each time you activate the remote journal, *ATTACHED is specified as the point in the receiver chain to start receiving journal entries. The system checks the currently attached remote journal receiver for journal entries and replicates the next journal entry in sequence.

You must specify the delivery mode when activating the remote journal. If needed, the delivery mode can be different on each activation of the remote journal.

Change journal operations that attach a new receiver to the local journal on system JKLINT are performed by the remote journal function as required on the target system. The remote journal function attaches the associated receivers to the remote journal automatically. If the remote journal is being maintained synchronously, the change journal operation to attach a new receiver is essentially a coordinated operation between the source and target systems. If the remote journal is being maintained asynchronously, the change journal operation to attach a new receiver on the target system is performed differently. In this case, it is triggered when the journal entry with journal code 'J' and entry type 'PR' is received by the remote journal on the target system.

The hot-backup application apply continues to replay changes to the data replica as received or retrieved from the receivers associated with the remote journal.

If needed, you can delete the receivers that are associated with the local journal on JKLINT when each receiver is replicated to JKLINT2. Sharon can accomplish this by specifying automatic deletion of journal receivers or manually deleting the receivers on JKLINT.

You can save the receivers from JKLINT2. If necessary, you can use the receivers for recovery of the original data on system JKLINT at a later time.

See Delete journal receivers for more information.

Data replication recovery if JKLINT fails

Recovery for JKLINT and JKLINT2 is simpler than environments that involve hot-backup because the hot-backup application does not switch-over to the backup system. What prevents the complications is an assumption that the hot-backup application apply logic will not receive and replay unconfirmed journal entries to the data replica if system JKLINT2 loses communications with system JKLINT. Therefore, the data replica on system JKLINT2 can never get ahead of the data on system JKLINT. This greatly simplifies data synchronization.

Scenario: Hot-backup environment

In this scenario, the remote journaling environment uses a hot-backup application that causes JKLINT2 to replace JKLINT in the case that JKLINT has a failure.

A **hot-backup application** typically performs the following:

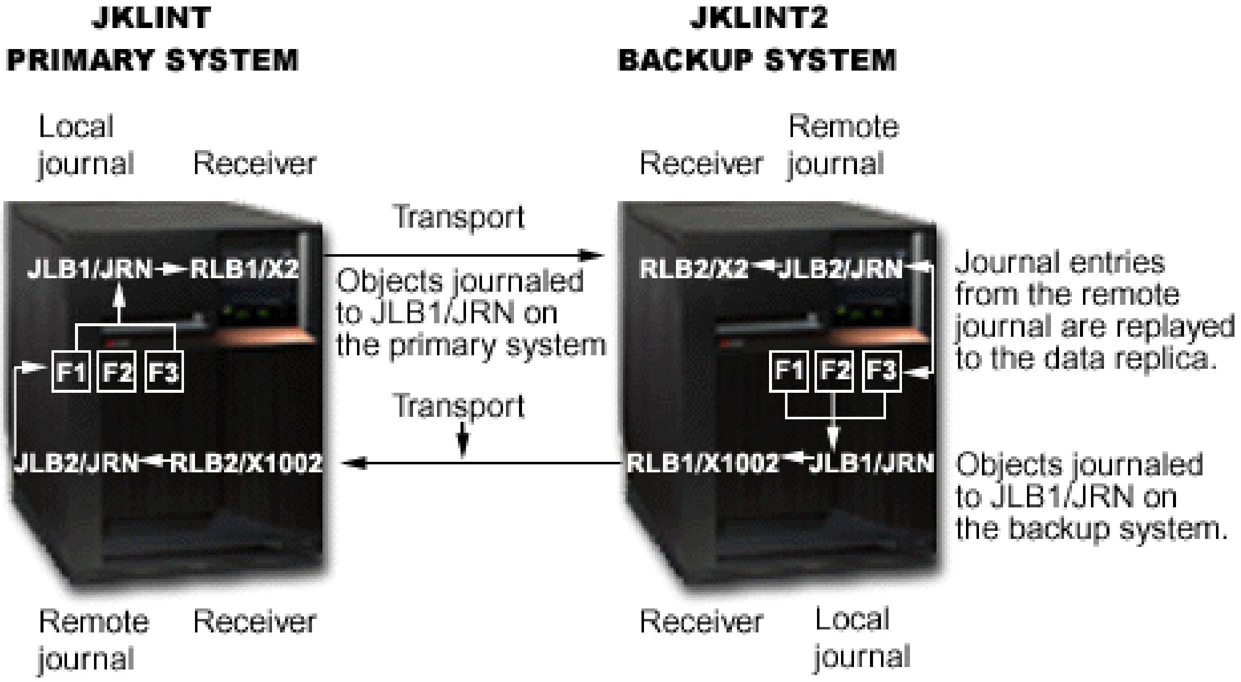
1. If the primary system fails, it performs a switch-over to the backup system. This function then logically promotes the backup system to assume the role of the primary system.
2. After the failed primary system is restarted, it performs a switch-back operation so that the primary system can again assume the role of the primary system.

A **hot-backup application apply** defines the part of a hot-backup application that actually performs the replay operations to the data replica. This usually occurs on the backup system when maintaining a data replica.

The following figure describes a typical remote journal environment that is used for hot-backup purposes. The following occurs in this illustration:

- Server JKLINT is the primary server while JKLINT2 is the backup server.
- Server JKLINT journals objects to local journal JKLB1/JRN.
- Changes to those journaled objects are also journaled to remote journal JLB2/JRN on server JKLINT2.
- On JKLINT2 a hot backup-apply replays changes to the data replica. When the hot backup-apply replays these changes, JKLINT2 journals the changes to its own local journal, JLB1/JRN.
- If JKLINT fails, JKLINT2 assumes the role of primary server and all local journaling of changes to the data replica (now acting as the original data) continue on JKLINT2's local journal, JLB1/JRN.
- When it is time to switch the role of primary server back to JKLINT, JKLINT2 sends changes from its local journal, JLB1/JRN, to remote journal JLB2/JRN on server JKLINT (the transport from JKLINT2 to JKLINT is only used for this purpose).
- JKLINT then uses its remote journal, JLB2/JRN, to replay changes to the original data.

Typical hot-backup environment with remote journal function



How to establish the hot-backup environment

The steps to establish a hot-backup environment the are the same as establishing data replication environment except for this additional last step:

> Sharon also establishes a remote journal JKLINT that is associated with the local journal that she creates on JKLINT2. This remote journal receives or retrieves the journaled changes that are made when JKLINT2 assumes the role of the primary system. This local journal and remote journal pair will only be used when replicating changes back to the original data. During normal run-time processing, the remote journal, JLB2/JRN, that is defined on JKLINT is not active. When it is not active, it is not receiving or retrieving journal entries from the local journal, JLB1/JRN, on JKLINT2

Normal run-time environment for the hot-backup environment

The details for run-time environment for the hot-backup environment is the same as the data replication environment.

Hot-backup recovery if JKLINT fails

If you use a hot-backup application where the logical ownership of the data is given to JKLINT2, recovery is more complicated. In this case, the hot-backup application logically promotes JKLINT to assume the role of the primary system. Recovery is more complicated because after JKLINT has completed its IPL, the remote journal function catch-up phase from the local journal on system JKLINT to the remote journal on system JKLINT2 will always allow a resynchronization of the two sets of data.

Data resynchronization is recovery processing that is performed during switch-back processing by a hot-backup application apply. This processing ensures that the original data is consistent with the data replica, and contains all the correct changes. The main objective of this, besides assuring data consistency, is to eliminate re-priming the original data from the data replica.

For details on recovering a hot-backup environment see Scenario: Recovery for remote journaling.

Scenario: Recovery for remote journaling

This scenario describes a hot-backup environment in which the local system, JKLINT fails. It is necessary to restore the local system, and synchronize it with the remote system, JKLINT2.

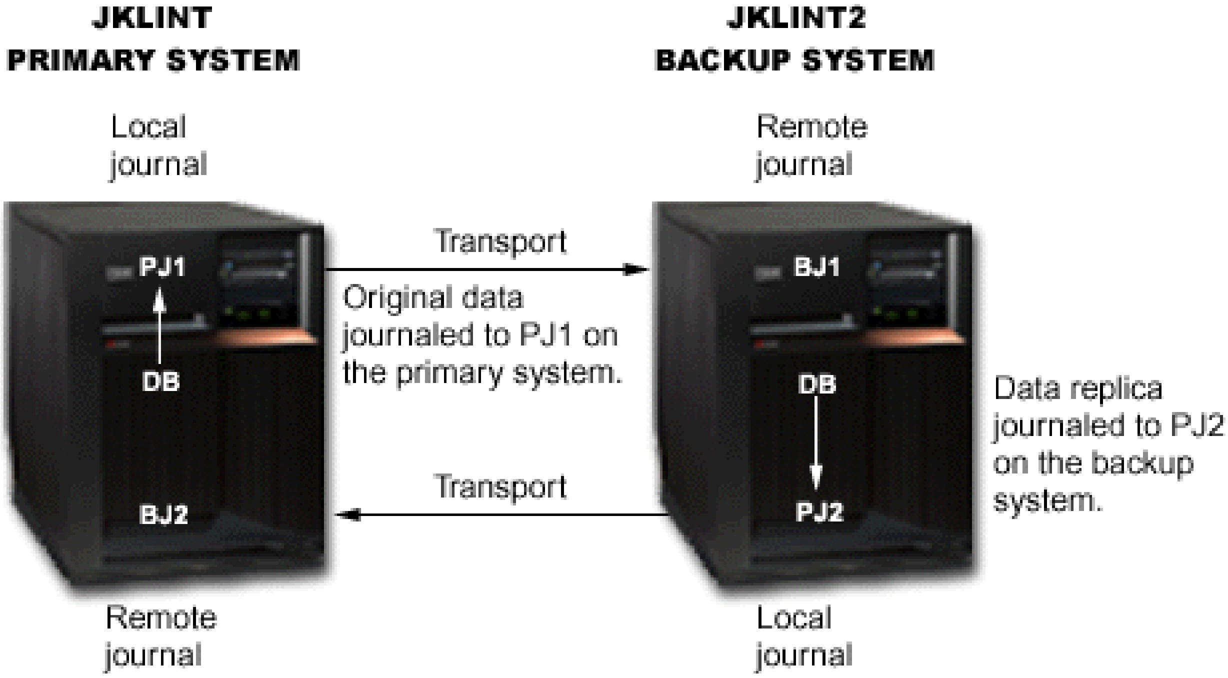
Details: Recovery for remote journaling scenario has step-by-step instructions for recovering from this failure this scenario describes.

This scenario, and the details for this scenario, only discuss database physical files. All the concepts, however, apply to any journaled object type.

Example remote journal environment for hot-backup recovery

The following figure illustrates the hot-backup environment for JKLINT and JKLINT2. The following items list considerations for this environment:

- The remote journal BJ2 is only active after JKLINT fails. JKLINT2 assumes the role of primary system and JKLINT is running again (as the secondary system).
- Journal receivers are not specifically called out in the figure. They have been omitted in an attempt to simplify the scenario and to focus on the recovery steps for the database. Where necessary, processing specific to journal receivers is referred to in the scenario.
- Likewise, library redirection for the journals and journal receivers is not specifically called out in the figure. Again, this is omitted in an attempt to simplify the scenario. In the scenario, the libraries for any of the journals or journal receivers could be redirected to a library that is different from that being used for the corresponding objects on the other system.
- The figure simply refers to the original data in the figure as DB on the primary system JKLINT and DB′ as the data replica on the backup system JKLINT2. DB can be one or more journaled objects, and DB′ contains a replica for each of the journaled objects in DB.



For simplicity, the scenario below treats DB as a single database file and DB′ as its replica.

The following items describe the scenario at the time JKLINT fails:

- System JKLINT is the primary system.
- The original data that is denoted by DB is journaled to an active local journal PJ1.
- Remote journal BJ1 on backup system JKLINT2 is active, and unless otherwise noted, is synchronously receiving journal entries from journal PJ1.
- A hot-backup application apply, not shown in the diagram, is asynchronously replaying, or applying, the changes to the data replica, DB'.
- The data replica DB' is journaled to local journal PJ2 on system JKLINT2.

  The journal state for journal PJ2 is *STANDBY.
- Remote journal BJ2 has a journal state of *INACTIVE (journal entries are not replicated to it). Remote journal BJ2 is only active when accepting the data changes back from system JKLINT2. This occurs after system JKLINT2 had been promoted to assume the role of the primary system due to a planned or unplanned outage of system JKLINT, and after system JKLINT has resumed operations.
- The primary system, JKLINT, has failed.
- The decision has been made to switch-over to the backup system, JKLINT2.

Details: Recovery for remote journaling scenario

These details provide a step-by-step description of the process that occurs in Scenario: Recovery for remote journaling

Current state of JKLINT and JKLINT2

At the time of the system failure, the state of JKL and JKLINT is as follows:

- Journal entries 12-19 are already deposited into PJ1 and confirmed in BJ1.
- The corresponding data changes are also already reflected in the data replica, DB', on system JKLINT2.
- Journal entries 20-25 are built and validated in main storage on JKLINT and sent to BJ1, and then system JKLINT fails.
- Main storage is not preserved when JKLINT fails, so at the time of the failure, the last known confirmed sequence number in BJ1 is 19. Sequence numbers 20 through 25 are all unconfirmed.
- The last known sequence number in PJ1 will be 19 the when system JKLINT restarts.

The hot-backup recovery strategy in these details does not require that both before-images and after-images are journaled to the local journal. However, the strategy would require before-images if, during the resynchronization process of the switch-back to the primary system, the strategy requires that the hot-backup application remove journaled changes. See step 3c.

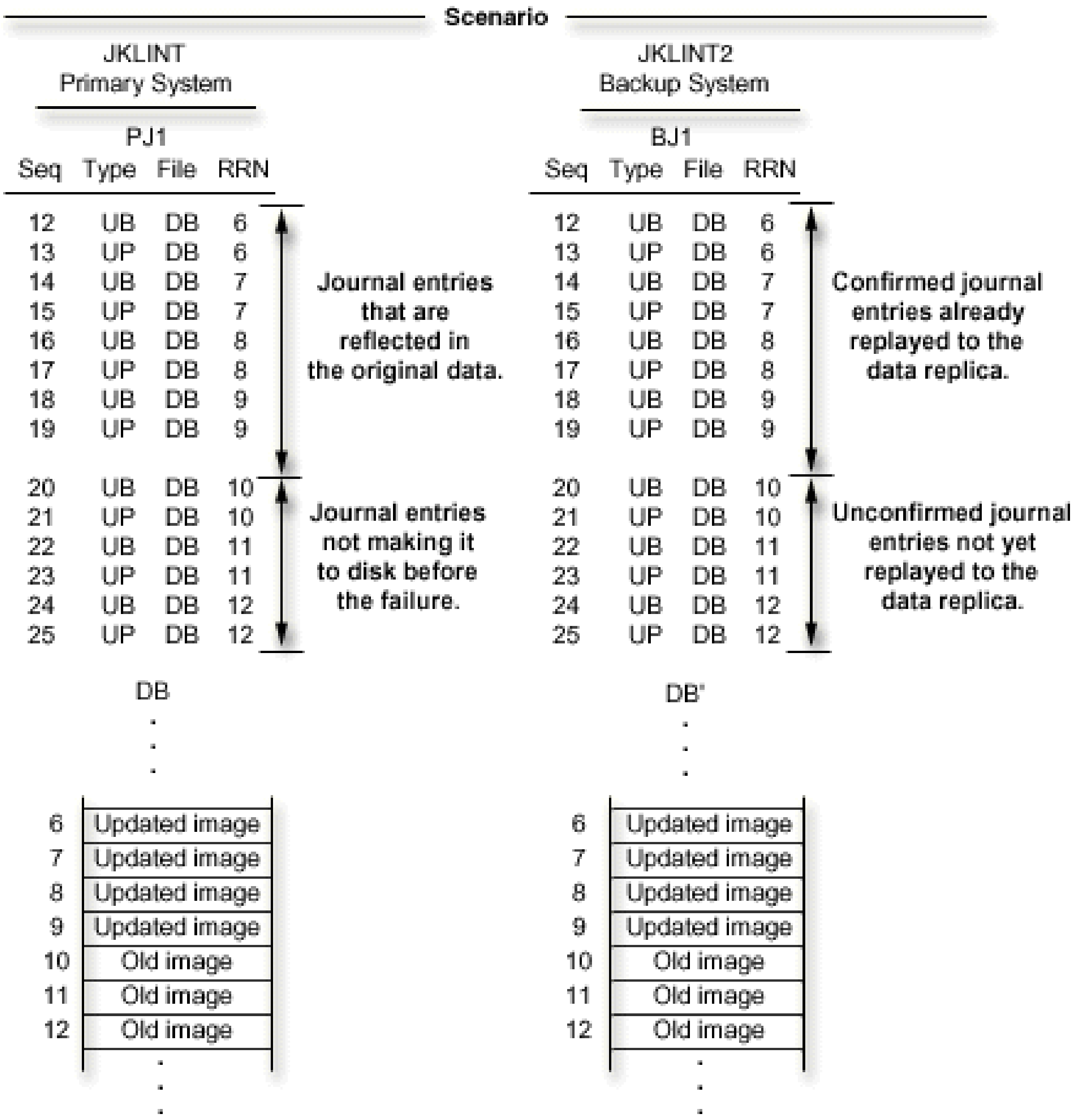
Steps required for recovery

To recover system JKLINT, the following steps are required:

1. **Update DB' by using the hot-backup application to replay the unconfirmed journal entries.**
   a. On system JKLINT2, allow the hot-backup application apply processing to complete the replay of confirmed operations as identified in journal BJ1. This is the first step of the switch-over processing. The apply processing includes replaying all journal entries up through and including sequence number 19.
   b. The hot-backup application does not replay sequence numbers 20-25 because the I/O for those journal entries is not yet confirmed from the local journal PJ1. The Receive Journal Entry (RCVJRNE) command or Retrieve Journal Entries (QjoRetrieveJournalEntries) API that is being used to retrieve the entries from the remote journal will not return sequence numbers 20-25 to the exit program, unless specifically requested to do so. To specify that sequence numbers 20 - 25 are returned to the exit program, use the INCENT(*ALL) parameter on the command. You can also request this by specifying *ALL for the include entries key on the API.
   c. After the hot-backup application replays all confirmed journal entries, perform a change journal operation to attach a new journal receiver to local journal PJ2 on system JKLINT2 and change the

state of journal PJ2 in *ACTIVE state. The change journal operation establishes a clean recovery point. It also makes clear what information needs to be sent back to system JKLINT later to replay back to the original data. Performing the change journal operation also prevents the remote journal function from having to re-replicate all of the journal entries that were previously generated into the currently attached journal receiver of PJ2. (The journal entries were generated into the receiver as part of replaying the database changes to the data replica on system JKLINT2.)

The following figure shows that more unconfirmed journal entries are present in BJ1 than are known in PJ1.


Scenario
JKLINT
Primary System
JKLINT2
Backup System
Journal entries that are reflected in the original data.
Journal entries not making it to disk before the failure.
Confirmed journal entries already replayed to the data replica.
Unconfirmed journal entries not yet replayed to the data replica.
DB
DB'

PJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9
20	UB	DB	10
21	UP	DB	10
22	UB	DB	11
23	UP	DB	11
24	UB	DB	12
25	UP	DB	12

BJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9
20	UB	DB	10
21	UP	DB	10
22	UB	DB	11
23	UP	DB	11
24	UB	DB	12
25	UP	DB	12

DB

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Old image
11	Old image
12	Old image

DB'

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Old image
11	Old image
12	Old image

2. **Perform switch-over processing and prepare JKLINT2 to run applications**
   a. The hot-backup application reads unconfirmed journal entries from BJ1 and replays them to the data replica. They are retrieved from BJ1 by using the Receive Journal Entry (RCVJRNE) command or QjoRetrieveJournalEntries API, specifically requesting that unconfirmed journal entries be returned. Journal entries 140-145 are generated into journal PJ2 when replaying these changes to the data replica.

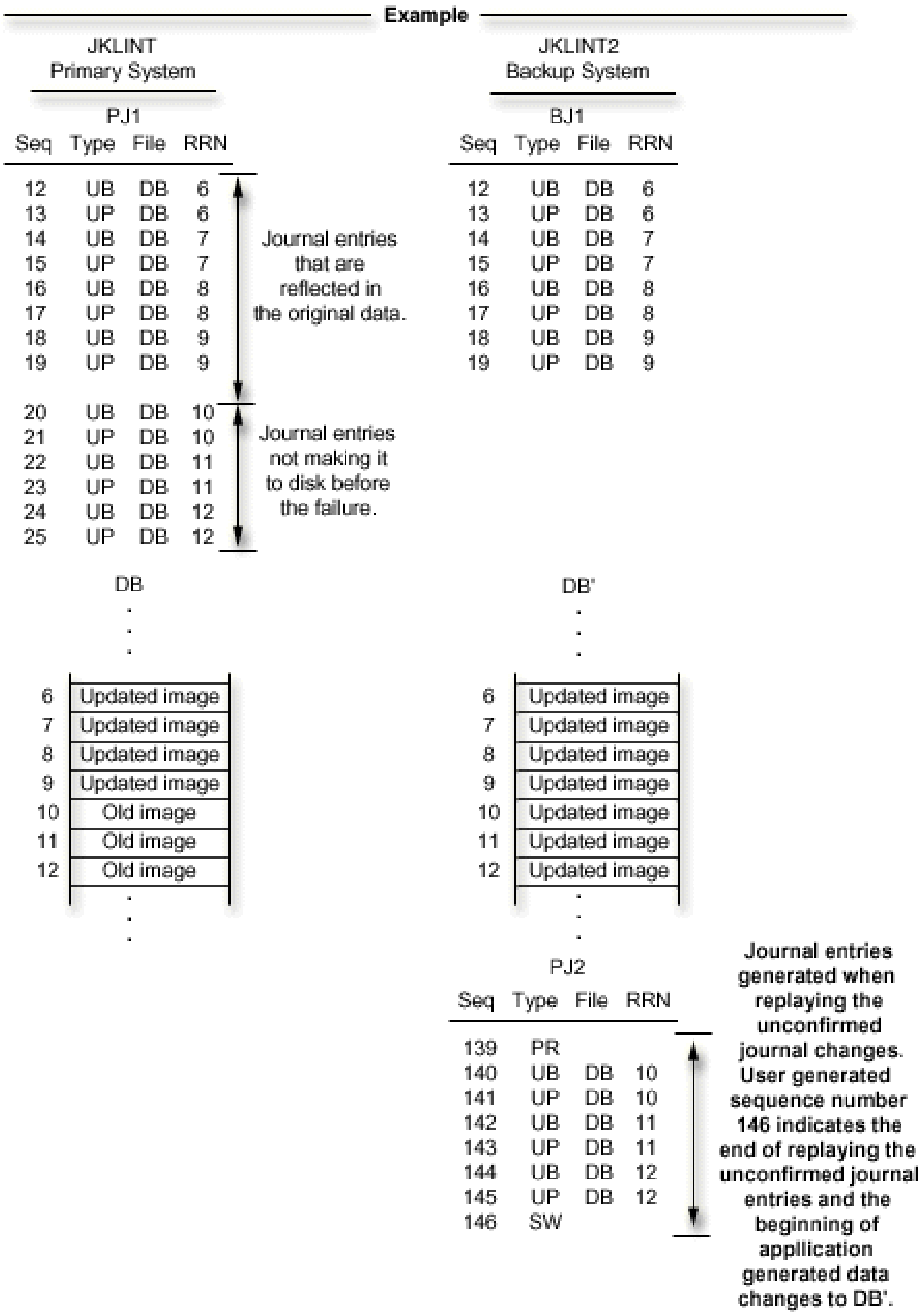
b. The QjoChangeJournalState API or CHGJRN command inactivates the remote journal BJ1. During this operation, the system physically removes the unconfirmed journal entries from BJ1. The last known sequence number in BJ1 is now 19.

c. The replay processing on JKLINT2 sends a user entry that indicates the point in time when the database was switched-over. The user entry in the following figure is sequence number 146, journal code ’U’, entry type ’SW’.

d. After these steps are performed on system JKLINT2, applications can now be started on JKLINT2 and use DB’ as the database to be updated. Applications continue to work and deposit journal entries 147-200.

e. System JKLINT restarts and normal IPL recovery finds the end of the journal for PJ1 to be sequence number 19. IPL recovery ensures that all changes up to sequence number 19 are reflected in the original data. The IPL for JKLINT completes with journal PJ1 being left in the *ACTIVE state, as this was the state of the journal when the system failed.

The following figure shows the state of BJ1, PJ2, and DB’ when system JKLINT2 is ready to assume the role of the primary system.



Example

JKLINT
Primary System

PJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9
20	UB	DB	10
21	UP	DB	10
22	UB	DB	11
23	UP	DB	11
24	UB	DB	12
25	UP	DB	12

Journal entries that are reflected in the original data.

Journal entries not making it to disk before the failure.

JKLINT2
Backup System

BJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9

DB

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Old image
11	Old image
12	Old image

DB'

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Updated image
11	Updated image
12	Updated image

PJ2

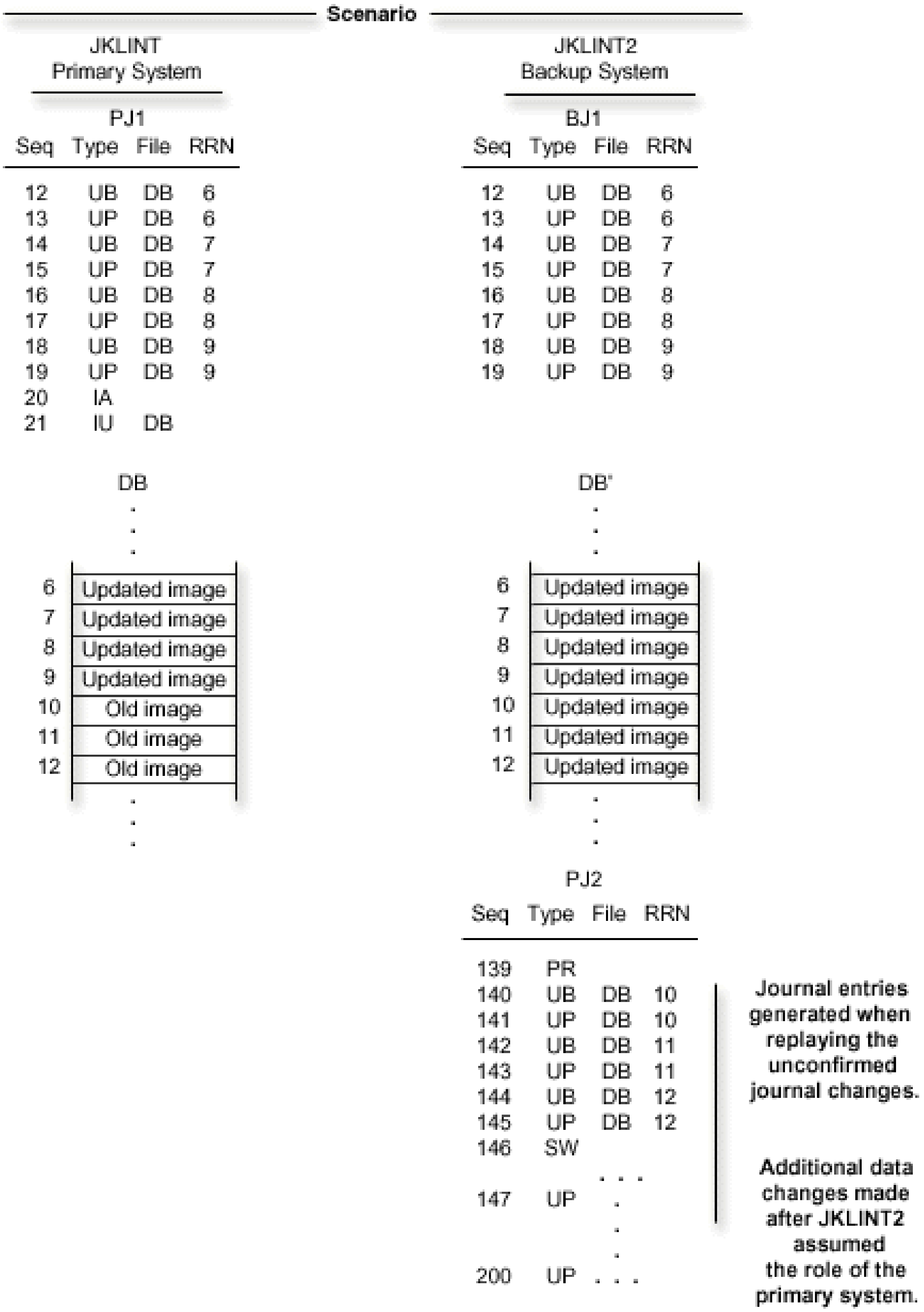
Seq	Type	File	RRN
139	PR		
140	UB	DB	10
141	UP	DB	10
142	UB	DB	11
143	UP	DB	11
144	UB	DB	12
145	UP	DB	12
146	SW		

Journal entries generated when replaying the unconfirmed journal changes. User generated sequence number 146 indicates the end of replaying the unconfirmed journal entries and the beginning of appllication generated data changes to DB'.

3. **Activate remote journal PJ2 and transport journal to JKLINT**
   a. After JKLINT restarts, activate the remote journal BJ2. Specify that the process will start with the attached journal receiver on JKLINT2. This starts the transport of journal entries representing the changes made on JKLINT2 as part of replaying the unconfirmed journal entries plus all changes made to DB′ while JKLINT was unavailable. While this transfer is progressing (during catch-up processing, which then transitions into synchronous or asynchronous remote journal function mode), changes are still being made by applications to DB′.
   b. Either before or during the transport of journal entries to BJ2, send and make known the last known sequence number in BJ1 (19) to the hot-backup application apply. This can be included as information in the SW user journal entry. See step 2c.
   c. The hot-backup application backs-out changes that are known to PJ1 (after the last known sequence number in BJ1) from the original data DB on system JKLINT. For this particular scenario, no changes need to be backed out of the original data.

Note: For scenarios which require this back-out processing, both before-image and after-image journal entries are required.

The following figure shows the state of both systems after system JKLINT has completed its IPL. This is after system JKLINT2 has been running as the primary system, but before database DB is resynchronized with DB′. (The database changes represented in PJ2 by journal sequence numbers 147-200 are not shown in DB′ for simplicity.)



PJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9
20	IA		
21	IU	DB	

BJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9

DB

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Old image
11	Old image
12	Old image

DB'

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Updated image
11	Updated image
12	Updated image

PJ2

Seq	Type	File	RRN
139	PR		
140	UB	DB	10
141	UP	DB	10
142	UB	DB	11
143	UP	DB	11
144	UB	DB	12
145	UP	DB	12
146	SW		
		. . .	
147	UP	.	
		.	
		.	
200	UP	. . .	

Journal entries generated when replaying the unconfirmed journal changes.

Additional data changes made after JKLINT2 assumed the role of the primary system.

4. **Replay changes to DB on JKLINT**

a. The hot-backup application replays the changes back to the original data on system JKLINT. The changes that are replayed include those changes that were made to DB' as part of the switch-over processing. The switch-over processing replayed the data changes for the unconfirmed journal entries (sequence numbers 140-145)). Additional changes include those data changes that were deposited while system JKLINT2 had assumed the role of the primary system (sequence numbers 147-300). Note that changes are still being made to DB' on system JKLINT2 and journal entries are still being generated into local journal PJ2 on system JKLINT2.

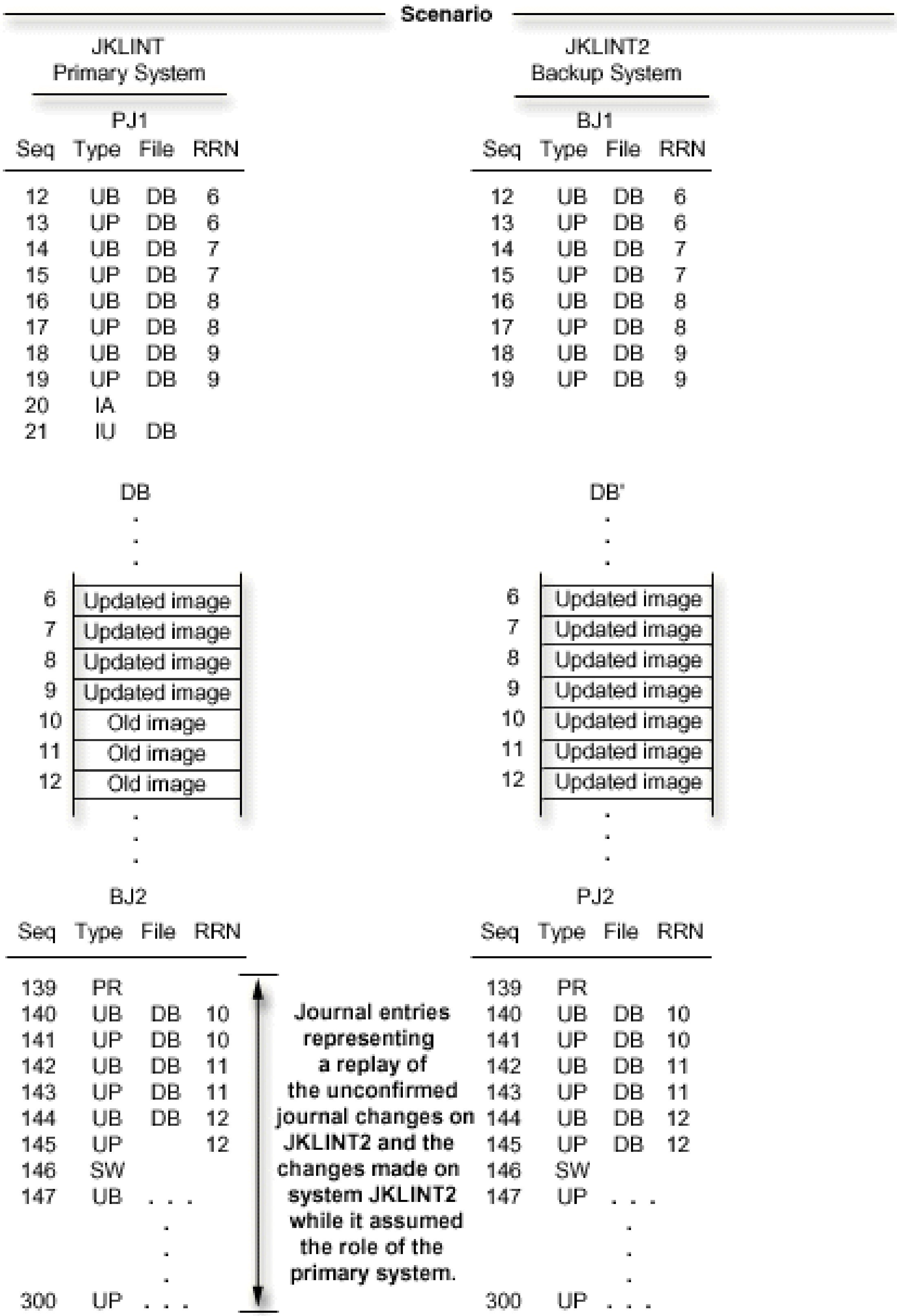
b. When you decide that JKLINT must again assume the role of the primary system, end the applications on JKLINT2. The following figure shows the state of both systems just before system JKLINT is going to assume the role of the primary system.

c. Allow the remaining changes to be replicated to BJ2. After all changes have been sent to BJ2, you can inactivate BJ2.

d. After all of the journal entries have been replayed to the original data on JKLINT, a attach a new journal receiver to PJ1 to clearly denote a new recovery point.

   The change journal operation is not absolutely essential. However, attaching a new journal receiver to PJ1 at this time makes clear where to start replaying changes back to the data replica on system JKLINT2. Performing the change journal operation also prevents the remote journal function from having to send back all of the journal entries that were previously generated into the currently attached journal receiver of PJ1. (The journal entries were generated in the receiver as part of replaying the data changes back to the original data on system JKLINT.)

The following figure shows the state of the journals and data just before starting to replay the changes back to the original data DB.



Scenario

JKLINT
Primary System

PJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9
20	IA		
21	IU	DB	

JKLINT2
Backup System

BJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9

DB

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Old image
11	Old image
12	Old image

DB'

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Updated image
11	Updated image
12	Updated image

BJ2

Seq	Type	File	RRN
139	PR		
140	UB	DB	10
141	UP	DB	10
142	UB	DB	11
143	UP	DB	11
144	UB	DB	12
145	UP		12
146	SW		
147	UB	. . .	
300	UP	. . .	

PJ2

Seq	Type	File	RRN
139	PR		
140	UB	DB	10
141	UP	DB	10
142	UB	DB	11
143	UP	DB	11
144	UB	DB	12
145	UP	DB	12
146	SW		
147	UP	. . .	
300	UP	. . .	

Journal entries representing a replay of the unconfirmed journal changes on JKLINT2 and the changes made on system JKLINT2 while it assumed the role of the primary system.

5. **Allow JKLINT to again assume role of the primary system**
   a. Application programs can now make changes to the original data DB on system JKLINT.

b. When you determine that it is time to start replicating the changes made on the primary system to the backup system, you can activate the remote journal BJ1.

   When activating the remote journal, you can indicate to send journal entries starting with the attached journal receiver on the source system. If this occurs, then only those journal entries that are required to be replayed to the data replica will be sent to system JKLINT2.

Note: You can start with the attached receiver, only if you did the change journal to attach a new receiver that was mentioned in step 4d.

c. If you want the complete chain of journal receivers from system JKLINT on JKLINT2, when you activate the remote journal, indicate to start with the attached journal receiver as known to the remote journal, BJ1. This will complete the sending of the journal receiver that contains the IPL entry (sequence number 20). The process will then move on to the next journal receiver that contains the journal entries where the hot-backup application apply will start replaying changes to the data replica. An alternative to that approach is to save and restore the detached journal receiver to system JKLINT2.

d. You change the state of local journal PJ2 on system JKLINT2 to *STANDBY state.

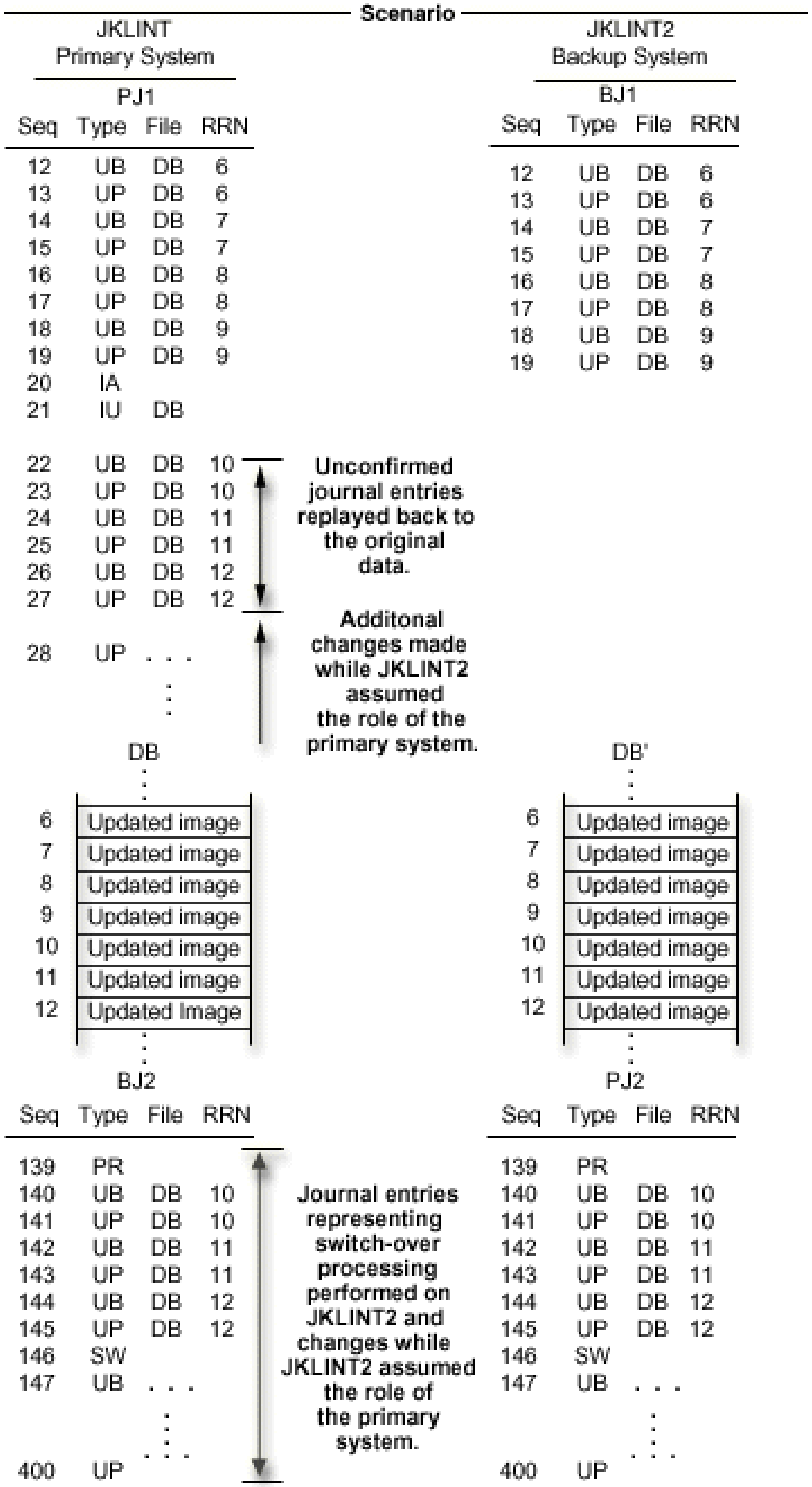
e. After local journal PJ2 has put in *STANDBY state, perform a change journal operation to attach a new journal receiver to PJ2.

   The change journal operation is not absolutely essential. However, attaching a new journal receiver to PJ2 at this time makes clear where the replaying of changes back to the data replica started on system JKLINT2. Performing the change journal operation also avoids the remote journal function from having to later send all of these hot-backup application apply generated journal entries back to system JKLINT.

   The newly attached journal receiver contains journal entries that will not have to be sent back to system JKLINT.

f. After the operation is performed, the hot-backup application apply can be started on system JKLINT2 to start replaying changes to the data replica. The hot-backup application apply starts with the source system sending the newly attached journal receiver.

The following figure shows that JKLINT is preparing again assume the role of the primary system.



Scenario

JKLINT
Primary System

PJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9
20	IA		
21	IU	DB	
22	UB	DB	10
23	UP	DB	10
24	UB	DB	11
25	UP	DB	11
26	UB	DB	12
27	UP	DB	12
28	UP	. . .	

Unconfirmed journal entries replayed back to the original data.

Additonal changes made while JKLINT2 assumed the role of the primary system.

JKLINT2
Backup System

BJ1

Seq	Type	File	RRN
12	UB	DB	6
13	UP	DB	6
14	UB	DB	7
15	UP	DB	7
16	UB	DB	8
17	UP	DB	8
18	UB	DB	9
19	UP	DB	9

DB

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Updated image
11	Updated image
12	Updated Image

DB'

6	Updated image
7	Updated image
8	Updated image
9	Updated image
10	Updated image
11	Updated image
12	Updated image

BJ2

Seq	Type	File	RRN
139	PR		
140	UB	DB	10
141	UP	DB	10
142	UB	DB	11
143	UP	DB	11
144	UB	DB	12
145	UP	DB	12
146	SW		
147	UB	. . .	
400	UP	. . .	

Journal entries representing switch-over processing performed on JKLINT2 and changes while JKLINT2 assumed the role of the primary system.

PJ2

Seq	Type	File	RRN
139	PR		
140	UB	DB	10
141	UP	DB	10
142	UB	DB	11
143	UP	DB	11
144	UB	DB	12
145	UP	DB	12
146	SW		
147	UB	. . .	
400	UP	. . .	

Related information for journal management

Listed below are the iSeries(TM) manuals and IBM(R) Redbooks(TM) (in PDF format) and Web sites that relate to the Journal management. You can view or print any of the PDFs.

Manuals

- AnyMail/400 Mail Server Framework Support (623 KB)
- Backup and Recovery (4 MB)
- CL programming (3.6 MB)
- iSeries Security Reference (6 MB)
- OptiConnect for OS/400(R) (868 KB)
- Performance Tools for iSeries (1.9 MB)
- Simple Network Management Protocol (SNMP) Support (391 KB)
- SNA Distribution Services (2.2 MB) on the V5R1 Supplemental Manuals Web site.
- TCP/IP Configuration and Reference (592 KB)
- Work Management (2.7 KB) on the V5R1 Supplemental Manuals Web site.
- WebSphere(R) Development Studio: ILE C/C++ Programmer's Guide (2.1 MB)

Redbooks

- Striving for Optimal Journal Performance on DB2 Universal Database(TM) for iSeries(TM) (3.1 MB)
- AS/400(R) Remote Journal Function for High Availability and Data Replication (1 MB)

Web site

DB2(R) UDB for iSeries Coding examples

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As...**
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

Appendix. Notices

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

IBM(R) 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 give you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY  10594-1785
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, 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:

© Copyright IBM Corp. 2002, 2004

```
IBM Corporation
Software Interoperability Coordinator, Department 49XA
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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, 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.

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. 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 IBM's application programming interfaces.

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

Trademarks

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

Application System/400(R)
AS/400(R)
DB2(R)
DB2 Universal Database(TM)
e (logo)
IBM(R)
iSeries(TM)
Operating System/400(R)
OS/400(R)
WebSphere(R)
xSeries(R)
400(R)

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

SET(TM) and the SET Logo are trademarks owned by SET Secure Electronic Transaction(TM) LLC.

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

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

Terms and conditions for downloading and printing publications

Permissions for the use of the publications you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

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(R).

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 AND FITNESS FOR A PARTICULAR PURPOSE.

All material copyrighted by IBM Corporation.

By downloading or printing a publication from this site, you have indicated your agreement with these terms and conditions.

Code example disclaimer

IBM(R) 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.

All sample code is provided by IBM for illustrative purposes only. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

All programs contained herein are provided to you ″AS IS″ without any warranties of any kind. The implied warranties of non-infringement, merchantability and fitness for a particular purpose are expressly disclaimed.

Printed in USA