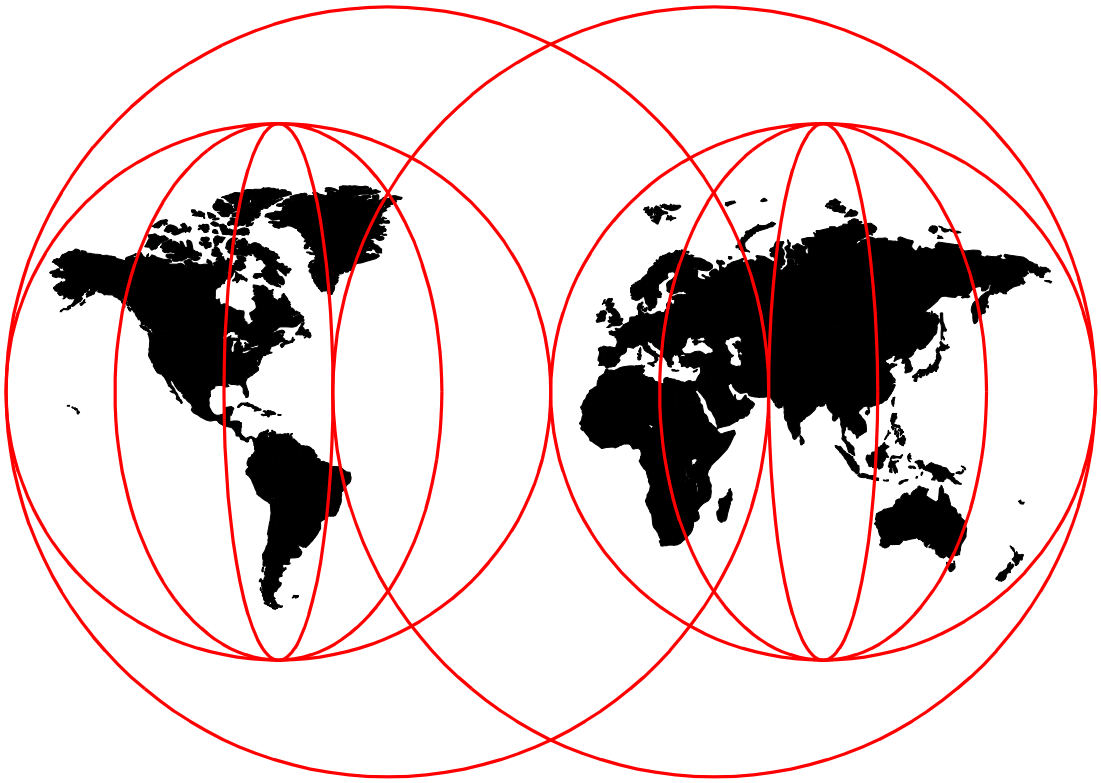
AS/400 Remote Journal Function for High Availability and Data Replication

Jarek Miszczyk, Frank Benson, Jens Grahn,
Patrick Sheehy, Gene Schumacher, Caleb KK Tang



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5189-00

International Technical Support Organization

SG24-5189-00

AS/400 Remote Journal Function for High Availability and Data Replication

February 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 103.

First Edition (February 1999)

This edition applies to Version 4, Release 3 of the IBM Operating System/400 (5769-SS1)

Disclaimer

This redbook contains performance values measured in laboratory environments with a specific benchmark. This information is presented along with general recommendations to help you have a better understanding of remote journal function. Your actual performance experience may differ substantially from these values. The benchmark results contained in this publication have not been submitted to any formal IBM test and are distributed *as is.*

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

Contents

Figures

Tables

Preface

Discover the finer details of AS/400 remote journal support. This redbook provides suggestions, guidelines, and practical examples about when and how to efficiently use remote journal function provided by OS/400. It contains information that you may not find anywhere else, including detailed coverage of the following topics:

- Synchronously and Asynchronously maintained remote journals
- Remote journal setup, removal, and day-to-day operation
- Remote journal performance over a wide range of communications gear
- Building AS/400 switchover solutions
- Guidelines for Independent Software Vendors (ISVs) that want to implement AS/400 high availability solutions

In particular, the remote journal performance chapter covers a wide range of test scenarios used to evaluate the following aspects of remote journal support:

- The efficiency of different communications hardware (OptiConnect, ATM, 100Mbps Ethernet, Token-Ring, and Frame Relay)
- The overall impact of running remote journal in both synchronous and asynchronous modes in an interactive transaction processing environment

This redbook is intended for system administrators, high availability specialists, and ISVs. It specifically focuses on their needs to implement a high availability solution for the AS/400 system. However, prior to reading this book, you should be familiar with journal function concepts.

By reading this redbook you gain a broad understanding of OS/400 commands and APIs that you can use for remote journal setup and operation. This redbook offers you the information you need to create or implement well-performing AS/400 high availability solutions.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Rochester Center.

Jarek Miszczyk is an International Technical Support Organization Specialist at the ITSO Rochester Center. He writes extensively and teaches IBM classes worldwide in all areas of DB2/400. Before joining the ITSO more than one year ago, he worked in IBM Poland as a Systems Engineer. He has over 10 years experience in the computer field. His areas of expertise include cross-platform database programming, SQL, and Object Oriented programming.

Frank Benson is a Product Architect with Lakeview Technology, Inc. He has over 30 years experience with IBM systems, including two years of that in the High Availability field. He holds a BS degree in math from the University of Minnesota. He has written previously on the topic of High Availability for SAP R/3.

Jens Grahn is a AS/400 Technical Consultant in Sweden He has 8 years of experience in the AS/400 field. He has worked at IBM for two years. His areas of expertise include Hot-backup Solutions, and Year 2000.

Patrick Sheehy is a Manager of Product Development for Lakeview Technology, Inc. He has 12 years of experience in application and systems development, on the AS/400 platform, including eight years in the programming laboratory at IBM-Rochester. He holds a BS degree in Computer Science from Winona State University. His areas of expertise include high availability, document library services, and save and restore. He frequently speaks at COMMON on the topics of Document Library Objects and OfficeVision/400 Security.

Gene Schumacher is a Senior Programmer Analyst with Vision Solutions. He has 15 years of experience in application development on System 38 and AS/400 platforms. His areas of expertise include building High Availability solutions for AS/400 with CL and RPG programming languages.

Caleb KK Tang is an Advisory I/T Specialist in IBM China/ Hong Kong. He has seven years of experience in Application Development, mainly on the AS/400 platform. He holds a BS degree in Computer Science and Software Engineering from the University of Birmingham in the UK, and an MBA from the University of Melbourne in Australia. His area of expertise include application development, specializing in business packages.

Without the vision and expertise of the following people this book would not exist:

Bruce Hansel
Dave Hermsmeier
Peg Levering
Ron Peterson
Chad Olstad
Mike Schambureck
Judy Trousdell
Larry Youngren
Rochester Development

Special thanks goes to the Large Network Environment team in Rochester:

Teresa Dean
Ron Naze

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 117 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Introduction to Remote Journal Function

The addition of the remote journal function on the OS/400 in V4R2 offers a reliable and fast method to transfer journal receiver data to a remote AS/400 system. It is ideal for use in data replication or high availability environments.

Remote journal allows you to establish journals and journal receivers on the target system that are associated with specific journals and journal receivers on the source system. Once the remote journal function is activated, the source system continuously replicates journal entries to the target system.

The remote journal function is a part of the base OS/400 system and is not a separate product or feature. It is implemented at the Licensed Internal Code layer. The benefits of the remote journal function include:

- It lowers the CPU consumption on the source machine by shifting the processing required to receive the journal entries from the source system to the target system.
- It eliminates the need to buffer journal entries to a temporary area before transmitting them from the source machine to the target machine. This translates into less disk writes and greater DASD efficiency on the source system.
- Since it is implemented in microcode, it significantly improves the replication performance of journal entries and allows database images to be sent to the target system in realtime. This realtime operation is called the *synchronous delivery mode*. If the synchronous delivery mode is used, the journal entries are guaranteed to be in main storage on the target system prior to control being returned to the application on the source machine.
- It allows the journal receiver save and restore operations to be moved to the target system. This way, the resource utilization on the source machine can be reduced.

1.1 The Traditional Approach

Before describing the remote journal function, this section looks at the traditional approach for the high availability solution and its drawbacks.

The high availability solutions, available prior to V4R2, used local journaling and the Receive Journal Entry (RCVJRNE) command. In a traditional environment, a user's applications that run on a source (production) system generate database changes. In turn, these changes create journal entries written to a local journal receiver. Still on the source side, the entries are received from the journal and buffered in a communications staging area. The data is transmitted asynchronously to the target system using existing communications gear. A high availability application running on the target system receives the journal entries into a temporary storage location, usually a user space. Another job, or many jobs, replays the changes into a copy of the source database. At this point, you have an exact copy of the production database on the target machine.

This hot-backup scenario has two major drawbacks:

- The asynchronous mode of operation means data latency. Your source system may fail while there are some journal entries waiting for transmission to the target machine. When this occurs, a few final database transactions are trapped on the source system. The database replica on the target system lags behind the production database on the source system.
- Increased CPU utilization. The data passes through many layers of the system software. It crosses the Machine Interface (MI) boundary several times. This costs extra CPU cycles.

Figure 1 illustrates the traditional solution. The shaded boxes indicate parts of the high availability solution that you have to purchase or develop in your IT environment.

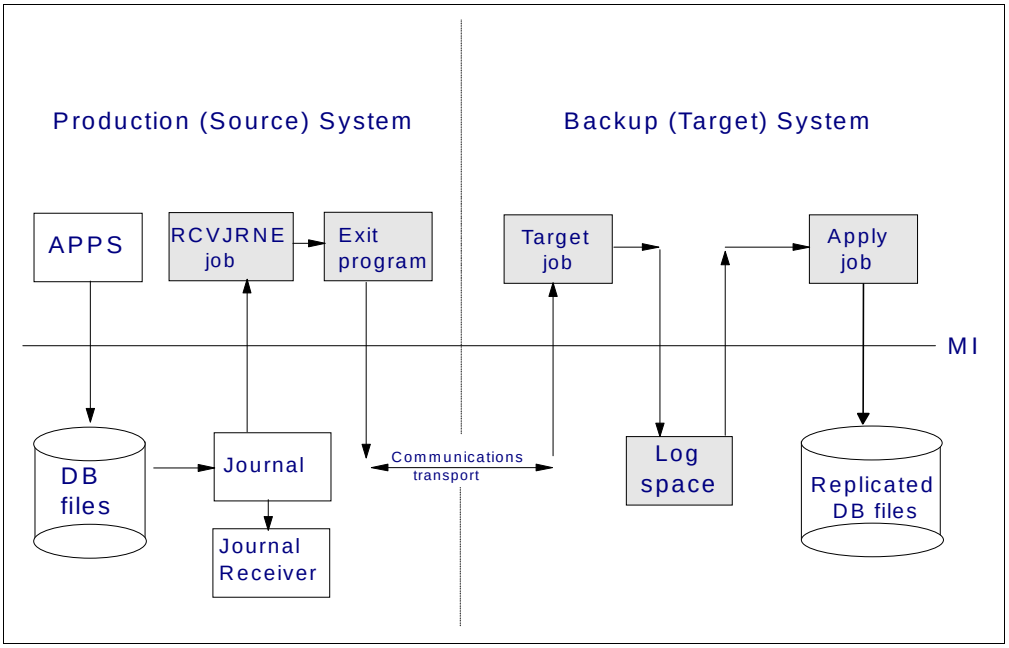


Figure 1. Example of a Traditional High Availability Solution

1.2 A Remote Journal Solution

The remote journal function provides a much more efficient transport of journal entries than the traditional approach. In this scenario, when a user application makes changes to a database file, there is no need to buffer the resulting journal entries to a staging area on the production (source) system. Efficient low-level system code is used instead to capture and transmit journal entries directly from the source system to associated journals and journal receivers on a target system. Much of the processing is done below the Machine Interface (MI). Therefore, more CPU cycles are available on the production machine for other important tasks. Because the remote journal function, if activated in synchronous mode, replicates journal entries to the backup machine's main storage before updating the production's system database, the data latency is driven to zero.

The high availability solutions available on the AS/400 system can fully take advantage of this more efficient transport mechanism. In fact, a high availability solution is still necessary in most customer environments to apply the

application-dependent data to a replica database for hot-backup scenarios. It is also necessary for providing the required management facilities for these hot-backup environments. In addition, the existing high availability products support replication of objects other than database files. Please refer to 5.6, “Other Considerations for Switchover” on page 91 for details.

Figure 2 illustrates a hot-backup solution based on the Remote Journal function. The shaded boxes indicate the parts of the high availability solution that you have to purchase or develop in your work area.

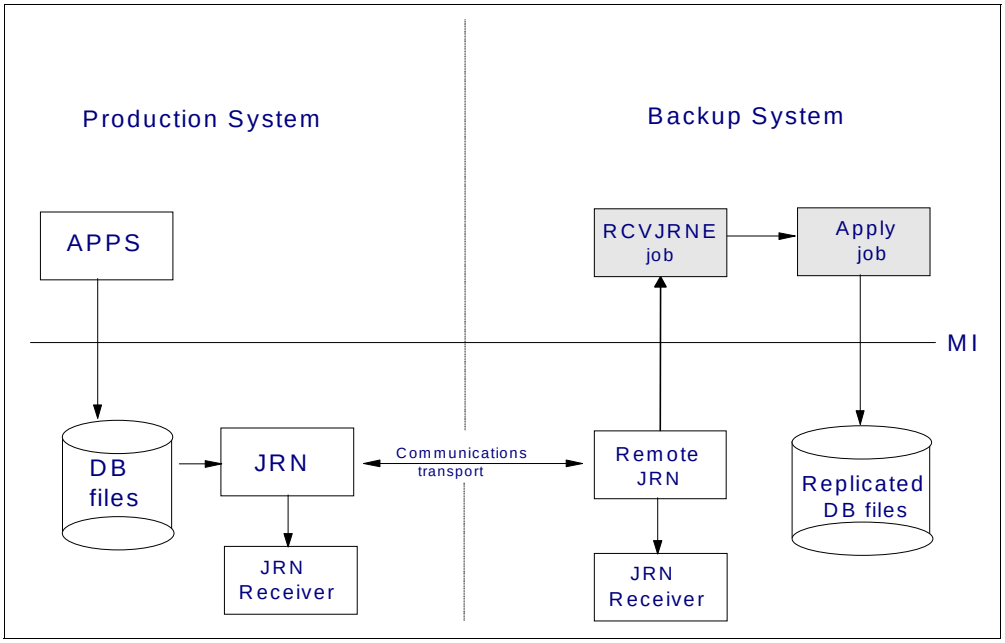


Figure 2. Example of a Hot-Backup Solution with Remote Journaling

1.3 Remote Journal Basics

When the remote journal function is activated on the source machine, the system replicates existing journal entries first as quickly as possible. This is referred to as *catch-up mode*. Once the specified journal receivers are transmitted to the target machine, the source starts continuously sending new entries either synchronously or asynchronously. The mode of operation depends on what was specified when the remote journal function was activated. The different delivery modes are discussed in the following sections.

You can consider the journal receivers on the target machine as a replica of the production machine’s journal receivers. It is as if you saved the production machine’s journal receivers and restored them on the target machine. The time stamps, system name, and qualified journal receiver names in the associated remote journal’s journal entries are exactly the same as in the local journal’s journal entries on the source system. In addition, the attach and detach times of the journal receivers are the same. However, while using remote journal support, you may see a minimal discrepancy in size between the local receiver and the associated remote receiver. Please note that you are using two different machines, which pre-allocate space for the journal receivers in different operating

system environments. This may result in slightly different sizes, but the data in the associated receivers is always the same.

1.3.1 Synchronously Maintained Remote Journals

In synchronous mode, journal entries are replicated to the main memory on the remote machine first. After an arrival confirmation is returned to the source machine, the journal entry is deposited to the local receiver. Next, the actual database update is made, if appropriate. The target system is updated in realtime with all of the journal entries as they are generated by a user application on the source system. Synchronous journaling allows for recovery that loses no journal entries on the target system if an outage is experienced on the source system. Sending journal entries synchronously to a target system modestly impacts the journaling throughput on the source system.

The main advantages of the synchronous delivery mode include:

- There will be no trapped transactions on the production system. The synchronous mode should definitely be your choice if it is intolerable to switchover to the target machine with a few of the final database transactions trapped on the source machine because these transactions were not sent to the target in realtime.
- The journal image reaches the target machine before reaching the disk of the source system. Therefore, there is no delayed arrival of journal entries.
- The efficient utilization of high speed communications hardware (such as OptiConnect and ATM). Refer to 4.4, “Remote Journal Performance Benchmarks” on page 73 for more details.

Figure 3 shows you a simple diagram of the tasks involved in synchronous mode. The shaded boxes show the parts of the high availability solution that you have to purchase or develop in your shop.

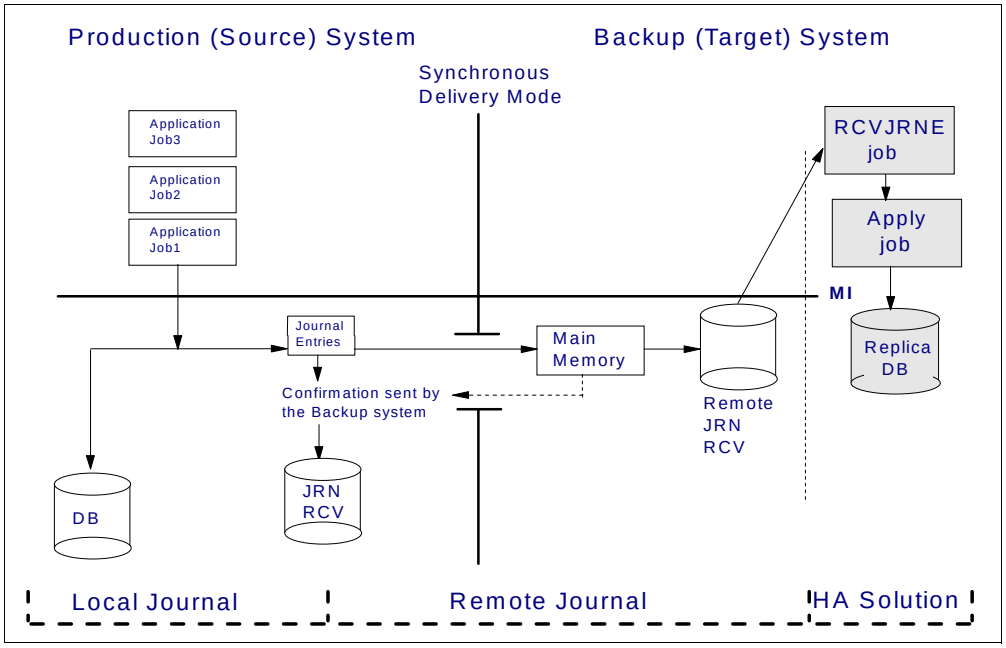


Figure 3. Synchronous Delivery Mode Tasks

1.3.2 Asynchronously Maintained Remote Journals

Sending journal entries asynchronously means that the journal entry is sent to the target system at some time after control is returned to the end user application that deposited the journal entry on the source system. From a recovery standpoint, asynchronous mode is less desirable. The reason is that the source system may have journal entries ahead of those journal entries that are known to the target system. Using this method allows for recovery that may lose a number of journal entries given a failure on the source system. It should have minimal impact to the local system when compared to the synchronous delivery mode.

The main advantage of the asynchronous delivery mode is the minimal impact on the production machine.

The disadvantages of the asynchronous delivery mode includes:

- A risk that some of the final database transactions may be trapped on the source system.
- In case of production machine's failure, the process of re-sychronizing the primary and replica databases is much more difficult when compared to the synchronous delivery mode.

However, asynchronous mode still remains a valid choice for those who intend to implement the remote journal function. You should consider asynchronous mode when:

- Your shop can tolerate the delayed arrival of journal entries, such as in data warehousing environments
- Your system is heavily utilized and you do not have resources to compensate for even very moderate overhead of synchronous mode
- You have slower communication gear that cannot keep up with the volume of journal traffic generated on your production machine

Refer to Chapter 4, "Remote Journal Performance" on page 69, for a detailed discussion on remote journal performance tested in different scenarios.

Figure 4 on page 6 shows a simplified diagram of tasks involved in asynchronous mode. The shaded boxes show the parts of the high availability solution that you have to purchase or develop for your environment.

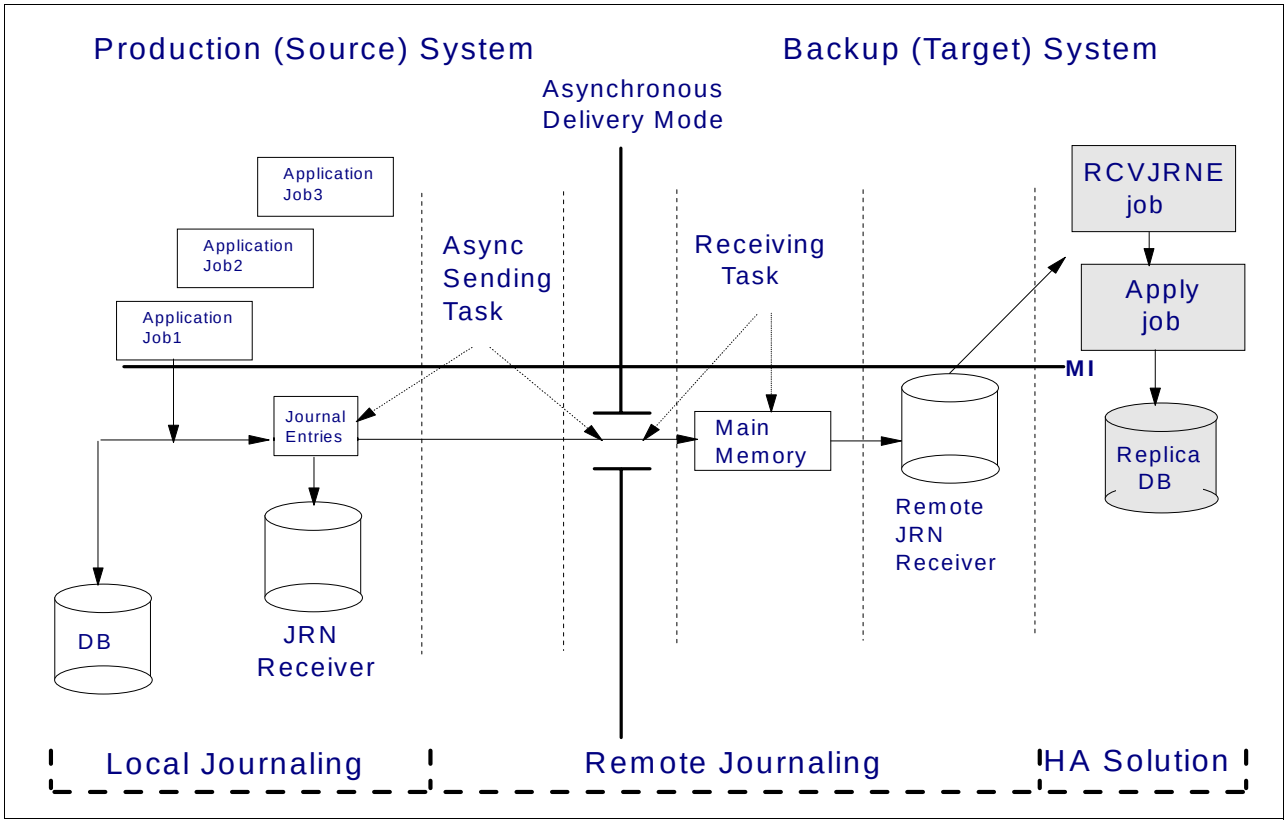


Figure 4. Asynchronous Delivery Mode Tasks

1.3.3 Remote Journal Configurations

There are two basic remote journal configurations, each of which are discussed in the following sections.

1.3.3.1 Broadcast Configuration

In a broadcast configuration, one AS/400 system replicates the local journal entries to one or many remote journals simultaneously. A broadcast configuration can run both in synchronous and asynchronous modes. The maximum broadcast configuration is limited to 255 remote systems. We recommend also that only one journal be activated in synchronous mode. Figure 5 on page 7 illustrates a broadcast configuration.

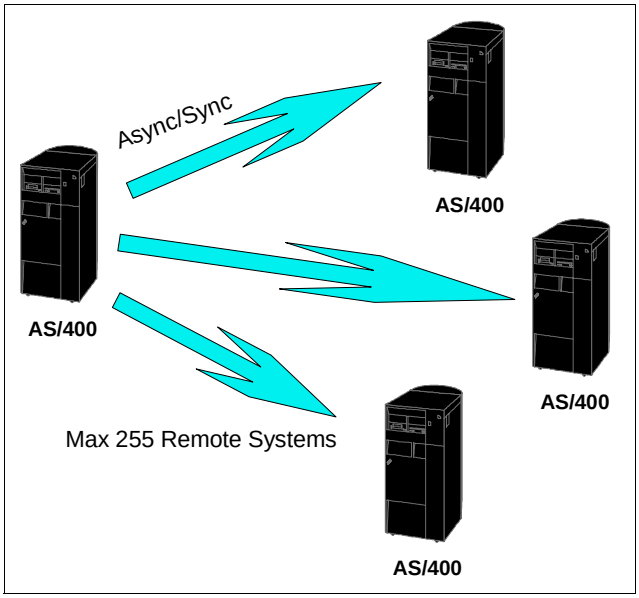


Figure 5. Broadcast Configuration

1.3.3.2 Cascade Configuration

In the cascade configuration, one AS/400 system replicates the journal entries to a remote journal on another AS/400 system. That remote journal can replicate the journal entries to yet another remote journal, and so on with no limit. The local journals on the source machine can be attached to the remote journals on the first target machine either in asynchronous or synchronous mode. All remote journals cascading to other remote journals must run in asynchronous mode.

A remote journal can receive entries from the source, and at the same time, send entries to another target. Figure 6 shows a cascade configuration.

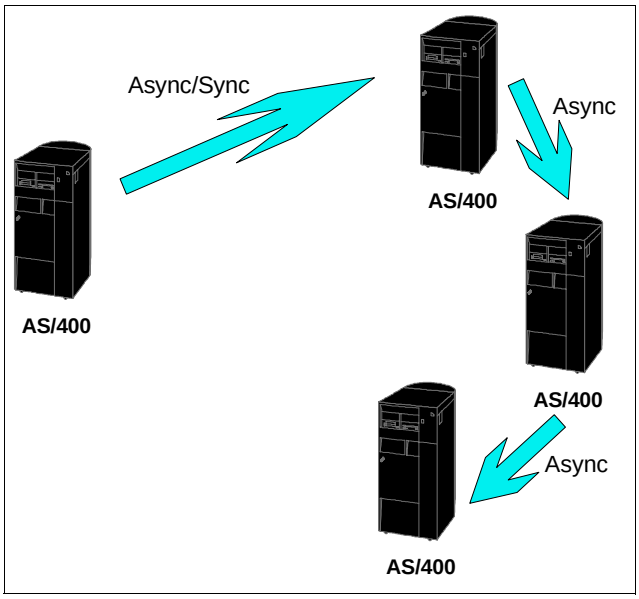


Figure 6. Cascade Configuration

A combination of broadcast and cascade configurations can be useful. If you want to reduce the amount of processing that is done on the production machine, you can replicate the journal entries to one remote journal on a target machine. That single journal can broadcast and cascade the journal entries to many other remote journals.

A local journal entry is deposited by an application. A remote journal always receives the journal entries from its associated local journal or another remote journal. You cannot directly deposit journal entries to a remote journal. They have to go through an associated journal on a source machine.

1.3.4 Remote Journal Types

There are two remote journal types: *TYPE1 and *TYPE2. Your decision on which one to choose depends on your save and restore strategy. Using *TYPE1 gives you greater flexibility for your save and restore scenarios, but limits your naming conventions. In comparison, using *TYPE2 gives you greater flexibility in naming your remote journals and receivers, but can limit your save and restore strategy.

The following list summarizes the naming conventions available for the *TYPE1 remote journal:

- The journal name for the remote journal must be the same as the source journal.
- The journal library name can be redirected to a single different library. All remote journals of *TYPE1, which are associated with the given local journal, must reside in the same named library.
- The receiver library name can be redirected to a single different library. All receivers attached to remote journals of *TYPE1 must reside in the same named library.

Figure 7 on page 9 illustrates the naming convention for a *TYPE1 remote journal, which cascades to another *TYPE1 remote journal.

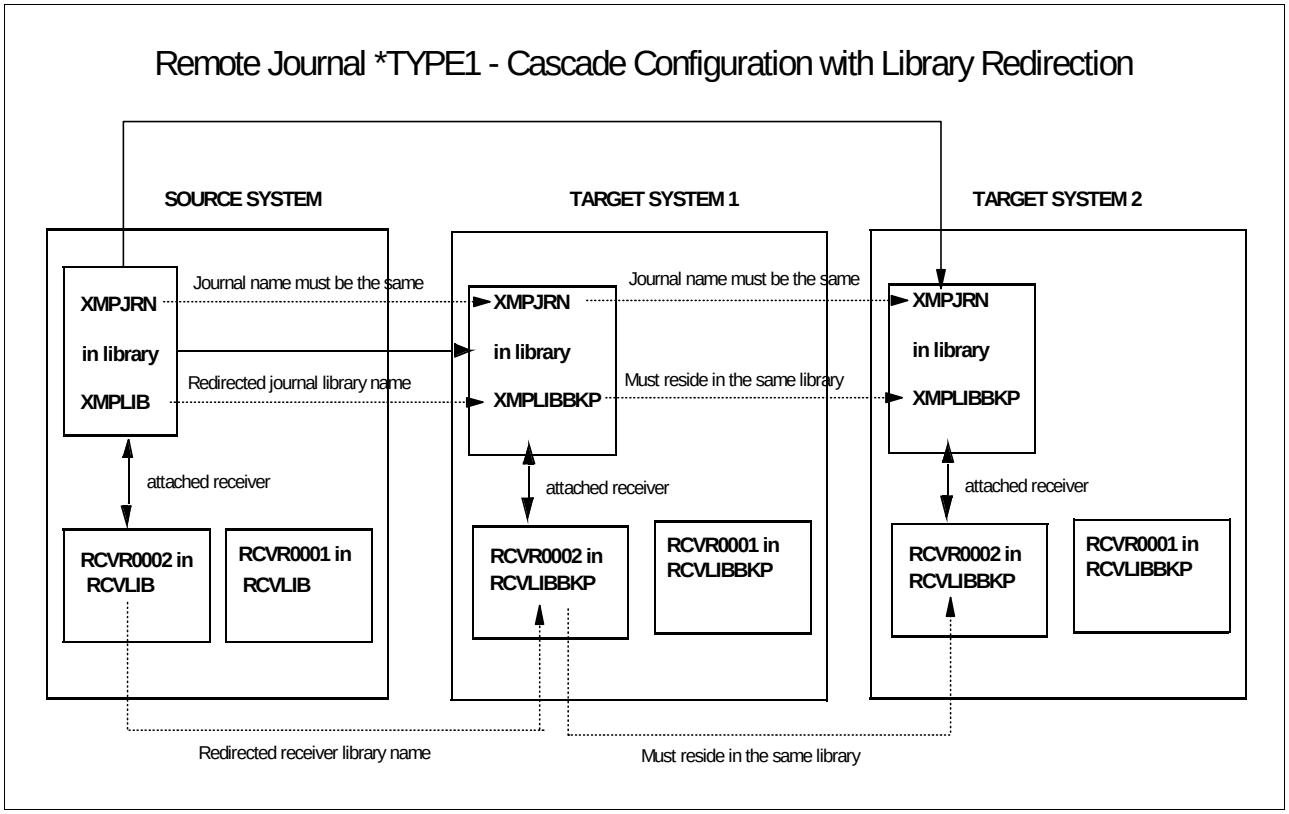


*Figure 7. Naming Convention for the Remote Journal *TYPE1*

The following list summarizes the naming conventions that are available for a remote journal *TYPE2:

- The journal name can be different from the source journal.
- There are no restrictions for journal library redirection.
- There are no restrictions for journal receiver library redirection.

Figure 8 on page 10 illustrates the naming convention for a *TYPE2 remote journal, which cascades to another *TYPE2 remote journal.

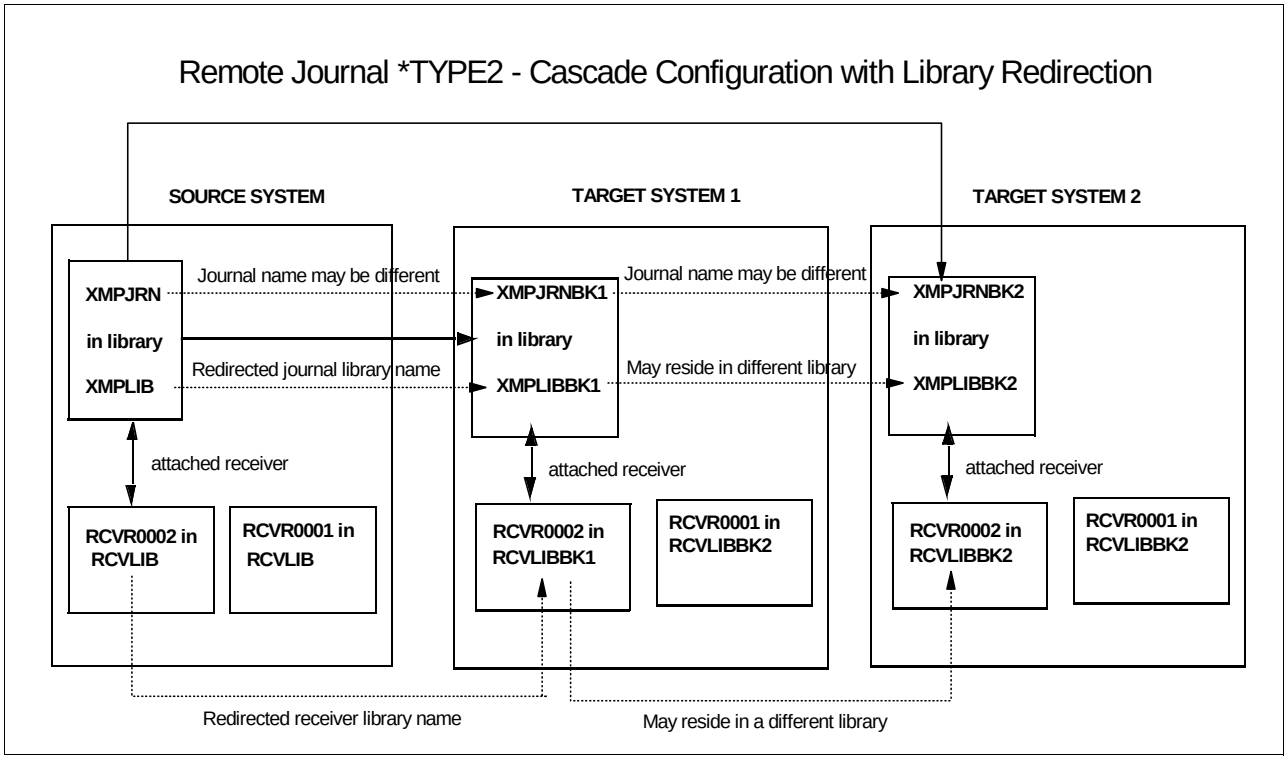


*Figure 8. Naming Convention for the Remote Journal *TYPE2*

The receivers associated with a *TYPE1 journal can be saved and restored to the source system or to any system where an associated *TYPE1 remote journal resides. The receivers associated with a *TYPE2 journal can be saved and restored to the source system where the local journal resides, or the same system where this particular *TYPE2 remote journal resides. If these restores are successful, the receivers are linked into the correct receiver chain of the journal. You can restore the receiver to another system, but it will not be associated with a remote journal on that system. It will only be associated with a local journal on that system.

Note

Remote journals of *TYPE1 and *TYPE2 can be mixed and matched. The library redirection scenarios shown in this section do not use mixed journal types for the sake of simplicity.

Refer to 3.8, “Save and Restore Considerations” on page 64 for a more detailed discussion on save and restore considerations for the remote journal function and the relationship to the remote journal types.

Chapter 2. Remote Journal Setup and Removal

This chapter describes how to set up the environment for the remote journal function, and how to handle the creation and removal of remote journals. The following topics are discussed:

- Remote journal function prerequisites
- Adding remote journals
- Removing remote journals
- Version considerations

Note

In this Redbook we use CL commands to manipulate remote journals. If your shop has not moved to V4R3 yet, please refer to the following URL for sample code on how to call the V4R2 remote journal APIs:
http://www.as400.ibm.com/db2/db2code.htm

Before the remote journal can be created and activated for replication, the following steps need to be performed:

1. Set up the communications path(s) that will be used to replicate journal entries between the source and target systems.
2. Create a Relational Database (RDB) Directory Entry that identifies the target system and communications link which will be used
3. Create local journals and receivers
4. Create and start the local journaling of physical files so that changes are recorded into the source journal.
5. Add the remote journal(s).

Figure 9 illustrates the processes of setting up the remote journal environment:

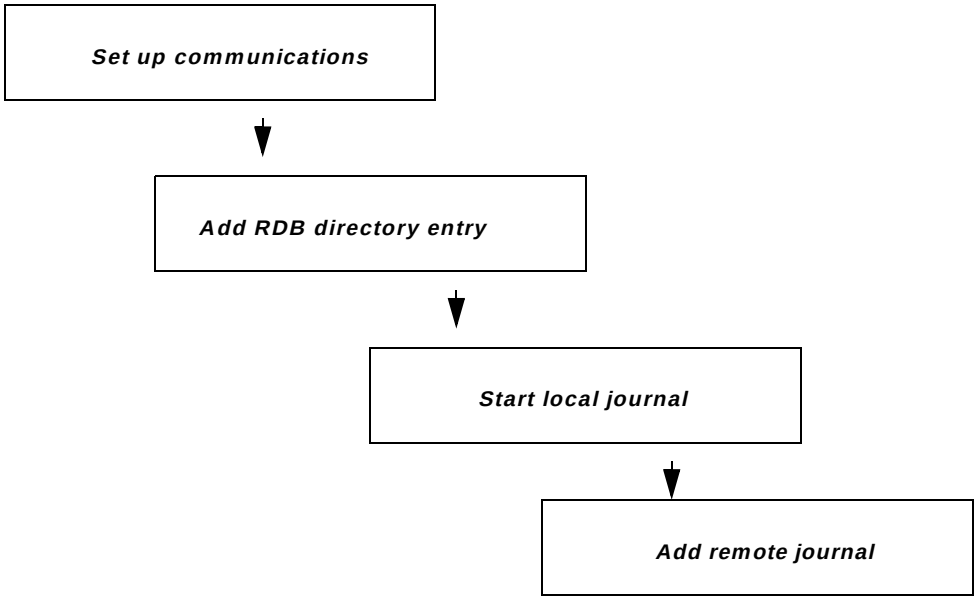


Figure 9. Steps to Setup a Remote Journal Environment

2.1 Remote Journal Function Prerequisites

Before the remote journal function can be established, certain features must be set up. There are also some considerations that will affect the performance of remote journal function.

2.1.1 Setting Up the Communications Environment and Security

The remote journal function requires a communication channel to be set up between the source and target system. Figure 10 shows the communications protocols that are supported by the remote journal function for replicating the journal entries to the remote systems.

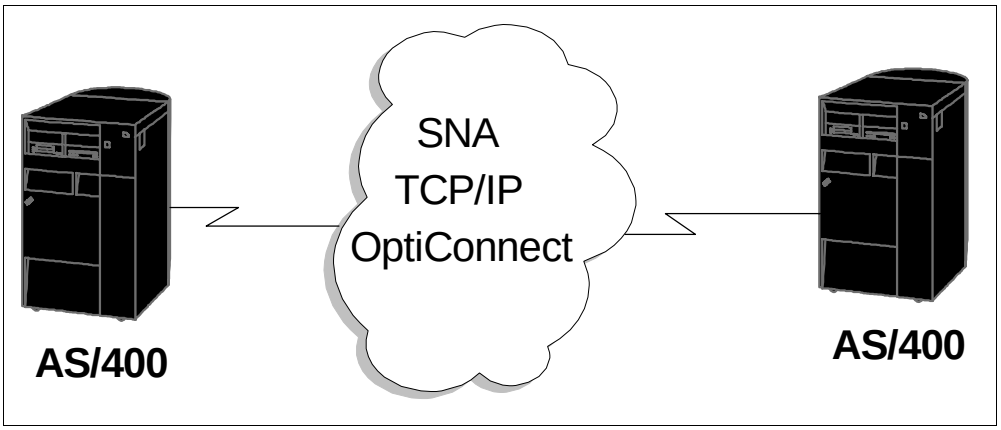


Figure 10. Different Communication Protocols Supported by Remote Journal Functions

- **OptiConnect for OS/400**

  If the OptiConnect for OS/400 bus transport method is used, the OptiConnect for OS/400 subsystem (QSOC) must be active on both the source system and the target system. The appropriate controllers and devices must be varied on.

  To view OptiConnect activity information and verify if OptiConnect is functional, you can use the Work with OptiConnect Activity (WRKOPCACT) command. The activity information shown represents cumulative activity from the collection start to the collection end times. In addition, you can also use the Verify OptiConnect Connections (VFYOPCCNN) command to verify connections to systems in the fiber optic network.

  Refer to *OptiConnect for OS/400*, SC41-5414, for more information.

- **Systems Network Architecture (SNA)**

  If SNA communications transport is to be used, you need to vary on the appropriate line, controller, and devices, and ensure that the subsystem QCMN is active on both systems.

  To verify if SNA is functional, you can use the Work with Configuration Status (WRKCFGSTS) command on controllers. This shows you whether the controllers are varied on and ready to use.

  Refer to *SNA Distribution Services*, SC41-5410, for more information.

- **Transmission Control Protocol/Internet Protocol (TCP/IP)**

  If TCP/IP is used, you must start it with the Start TCP/IP (STRTCP) command, including the distributed data management (DDM) servers.

  To verify if TCP/IP is functional, you can use Work with TCP/IP Network Status (NETSTAT) command option 3. This option can display the TCP/IP connections between a local system and a remote system.

  Refer to *TCP/IP Configuration and Reference*, SC41-5420, for more information.

To verify if the communications subsystems are active, you can use the Work with Subsystem (WRKSBS) command to show information for each subsystem in the system. Pressing F11 displays the status of each subsystem. Figure 11 shows the Work with Subsystem command display.

```
                          Work with Subsystems
                                                              System:   SRCSYS
Type options, press Enter.
  4=End subsystem   5=Display subsystem description
  8=Work with subsystem jobs

                    Total     Subsystem  Active
Opt  Subsystem   Storage (K)   Number     Jobs   Status
     QBATCH                0    024311        0   ACTIVE
     QCMN                  0    024315        7   ACTIVE
     QCTL                  0    024264        1   ACTIVE
     QINTER                0    024307        2   ACTIVE
     QSERVER               0    024299       12   ACTIVE
     QSOC                  0    024408        1   ACTIVE
     QSPL                  0    024297        0   ACTIVE
     QSYSWRK               0    024265       97   ACTIVE

                                                                      Bottom
Parameters or command
===>
F3=Exit   F5=Refresh   F11=Display pools   F12=Cancel
F14=Work with system status
```

Figure 11. WRKSYS Prompt Display

To vary on (activate) communication descriptions, you can use the Vary Configuration (VRYCFG) command or option 1 in the Work with Configuration Status (WRKCFGSTS) command. If the non-switched line descriptions are varied on, the appropriate controllers and devices attached to that line are also varied on. The WRKCFGSTS command also gives the status of each connection. For more information about working with the communication configuration status, refer to Chapter 6, "Distributed Relational Database Administration and Operation Tasks" in *OS/400 Distributed Database Programming*, SC41-5702. Figure 12 on page 14 shows the WRKCFGSTS command with the controller status display.

```
                    Work with Configuration Status                    SRCSYS
                                                                 10/06/98  10:06:32
Position to  . . . . .              Starting characters

Type options, press Enter.
  1=Vary on   2=Vary off   5=Work with job   8=Work with description
  9=Display mode status    13=Work with APPN status...

Opt  Description       Status               -------------Job--------------
    OPTICOMM          ACTIVE
       OPTICOMM       ACTIVE
    TCPIPCOMM         ACTIVE
       TCPIPCOMM      ACTIVE               QTCPIP      QTCP        024363
    SNACOMM           ACTIVE
       SNACOMM        ACTIVE




                                                                       Bottom
Parameters or command
===>
F3=Exit   F4=Prompt   F12=Cancel   F23=More options   F24=More keys
```

*Figure 12. WRKCFGSTS CFGTYPE(*CTL) Display with Controller Status*

Here are the types of network interfaces that are supported by the AS/400 system:

- Token-Ring networks
- Ethernet networks
- Asynchronous Transfer Mode (ATM) emulated local and wide area networks
- Frame relay networks
- Wireless networks
- Distributed Data Interface (DDI) networks

Refer to *OS/400 LAN, Frame Relay and ATM Support*, SC41-5404, and *OS/400 Communications Configuration*, SC41-5401, to learn how to configure and improve the performance of your local area network.

Different network interface options affect the performance of the remote journal function differently. Remote journal performance for different types of network interfaces is discussed in Chapter 4, “Remote Journal Performance” on page 69.

2.1.2 Setting Up a Relational Database Directory Entry

Whenever you try to add, change, or remove a remote journal, a relational database (RDB) directory entry needs to be specified. All the systems in the network must have their RDB directory set up with connection information. The RDB directory contains database names and values that are translated into communications network parameters. Specifying an RDB directory entry identifies the communications protocol and corresponding communications path information over which the remote journal function will work.

Each RDB entry also identifies the method of accessing the RDB. One entry for each remote RDB must be added. Each RDB name must be unique within the RDB directory and within the distributed network.

To add an RDB directory entry, you can use the Work with RDB Directory Entries (WRKRDBDIRE) command, then option 1 for add. Or, you can use the Add RDB Directory Entry (ADDRDBDIRE) command instead.

It is a good idea to make the RDB name the same as the system name or location name specified for this system in your network configuration. This can help you to identify a database name and correlate it to a particular system in your distributed RDB network, especially if your network is complex.

The relational database directory is not an AS/400 object so it cannot be saved and restored with Save Object (SAVOBJ) command. However, the files that make up the relational database directory are saved with Save System (SAVSYS) command. Therefore, the physical file that contains the relational database directory can be restored from the save media to your library. You can use the information from this file to re-create the relational database directory. Another approach is to save the relational database directory by creating an output file with Display Relational Database Directory Entry (DSPRDBDIRE) command with the OUTFILE parameter specified. This output file can then be used to add entries to the directory in case it has been corrupted. Refer to the *OS/400 Distributed Database Programming*, SC41-5702, for a sample CL program that reads the content of the output file created with DSPRDBDIRE command. This sample program also re-creates the entries with the Add Relational Database Directory Entry (ADDRDBDIRE) command.

For remote journal usage, the RDB entry cannot refer to a *LOCAL database, nor can an Application Requester program (*ARDPGM) be used. Refer to the following sections for more information.

Security of the remote journal function depends 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 activities. However, you may consider isolating the remote journal function activity to have the best performance.

Note: While setting up the remote journal environment on multiple systems, make sure that the user profile, used to connect to remote systems, exists in all the systems in the network.

2.1.2.1 Setting Up an RDB Directory Entry for SNA

Setting up an RDB directory entry for SNA is straightforward. If you are running over OptiConnect, the steps for setting up an RDB directory are similar to the SNA setup process described in this section. Refer to *OptiConnect for OS/400*, SC41-5414, for more information on this communications method. Once you are sure that the appropriate controllers and devices are created and varied on, use either the WRKRDBDIRE or ADDRDBDIRE commands to add the relevant entry. In Figure 13 on page 16, an RDB directory entry named TGTSYS is added to allow access to the remote system TGTSYS using SNA.

```
                    Add RDB Directory Entry (ADDRDBDIRE)

 Type choices, press Enter.

 Relational database  . . . . . RDB          > TGTSYS
 Remote location:                RMTLOCNAME
   Name or address  . . . . . .               > TGTSYS



   Type . . . . . .                           *SNA
 Text . . . . . . . . . . . . . TEXT          RDB Dir Entry to TGTSYS system
via SNA
 Device:                         DEV
   APPC device description  . . .             *LOC
 Local location . . . . . . . . LCLLOCNAME    *LOC
 Remote network identifier  . . . RMTNETID    *LOC
 Mode . . . . . . . . . . . . . . MODE        *NETATR
 Transaction program  . . . . . . TNSPGM      *DRDA

                                                                       Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 13. ADDRDBDIRE Prompt Display

The required parameters include:

- **Relational Database (RDB)**

  Specify the name of the RDB being added. Each name entered into the directory must be unique. Because RDBs communicate with each other over a distributed network, each name must also be unique among other RDBs on the network. The system can determine whether an RDB name is unique in the directory, but you must determine whether it is unique among RDBs on the network.

- **Remote Location (RMTLOCNAME)**

  Specify the name or address of the system on which the RDB is located. For the SNA implementation, you need to configure the controller that governs the communication between the local and the remote system.

  Then, you need to refer to this controller in the device description parameter. The first element of this parameter can take several forms. For the SNA protocol, the relevant values are:

  - SNA remote location name (LU name). Specify a maximum of eight characters for the remote location name. If this form is used, the second element of this parameter must be *SNA (the default).
  - SNA remote network identifier and remote location name separated by a period. Specify a maximum of eight characters for the remote location name, and a maximum of eight characters for the remote network identifier. If this form of the parameter is used, the second element of this parameter must be *SNA (the default).

The optional parameters include:

- **Text (TEXT)**

  Specify text that briefly describes the RDB directory entry. You can specify no more than 50 characters of text, enclosed in apostrophes.

- **Device (DEV)**

  Specify the name of the advanced program-to-program communications (APPC) device description on the local system that is used to access this RDB. The device description does not need to exist when the RDB directory entry is added.

- **Local Location Name (LCLLOCNAME)**

  Specify the local location name by which the local system is identified to the system on which the RDB is located. The local location name cannot be the same as the remote location name.

- **Remote Network Identifier (RMTNETID)**

  Specify the remote network identifier of the system on which the RDB is located. If this parameter is specified, the RMTLOCNAME parameter must be consistent with this RMTNETID parameter. If the RMTLOCNAME parameter specifies a network ID, these two parameters must match (or an error message are issued). If the RMTLOCNAME parameter does not specify any network ID, there is no possibility for conflict between these parameters.

- **Mode (MODE)**

  Specify the mode name that is used with the remote location name to communicate with the system on which the RDB is located. More information on mode names is in the APPC Programming book.

- **Transaction Program (TNSPGM)**

  Specify the name of the transaction program to use with the RDB entry.

- **Application Requester Driver (ARDPGM)**

  This parameter is not applicable for remote journal.

The Display RDB Directory Entry (DSPRDBDIRE) command, or option 5 on the WRKRDBDIRE display, can be used to view an existing RDB directory entry. The Change RDB Directory Entry (CHGRDBDIRE) command, or option 2 on the WRKRDBDIRE display can be used to change an existing RDB directory entry.

In Figure 14 on page 18, the RDB directory entry created in this section is reviewed using DSPRDBDIRE command.

```
                    Display Relational Database Detail

Relational database  . . . . . :   TGTSYS
Remote location:
  Remote location  . . . . . :   TGTSYS
    Type . . . . . :   *SNA
  Device description . . . . . :   *LOC
  Local location . . . . . :   TGTSYS
  Remote network identifier  . . :   *LOC
  Mode . . . . . :   *NETATR
  Transaction program  . . . . . :   *DRDA
Text . . . . . :   RDB Dir Entry to TGTSYS system via
   SNA









                                                                      Bottom
Press Enter to continue.

F3=Exit   F12=Cancel
```

Figure 14. DSPRDBDIRE Display with an SNA RDB Directory Entry Definition

2.1.2.2 Setting Up an RDB Directory Entry for TCP/IP

Setting up an RDB directory entry for the TCP/IP protocol involves a couple of additional steps when compared to SNA or OptiConnect. Before we outline the process we need to describe briefly the major concepts of Distributed Relational Database Architecture (DRDA). Refer to Section 6.10 in the *DB2/400 Advanced Database Functions*, SG24-4249, for details.

In the distributed relational database environment, the system running the application and sending the SQL requests across the network is called *Application Requester* (AR). Any remote system that executes SQL requests coming from the Application Requester is known as *Application Server* (AS).

The TCP/IP implementation of the Distributed Relational Database Architecture (DRDA) Application Server (AS) is based on multiple connection-oriented server jobs running in the QSYSWRK subsystem. The DRDA server jobs are defined by prestart job entries. Once the Application Requester (AR) connects to the listener at the AS, the listener issues a request to wake up a prestarted server job. The listener then passes the socket descriptor to the server job and any further communication occurs directly between the client application and server job. By observing this socket descriptor, you can determine if TCP/IP is functional.

Figure 15 on page 19 shows a simple form of a distributed relational database environment. The SRCSYS system acts as an AR to run the application and send requests across the network. The TGTSYS system acts as an AS to execute requests coming from the AR.

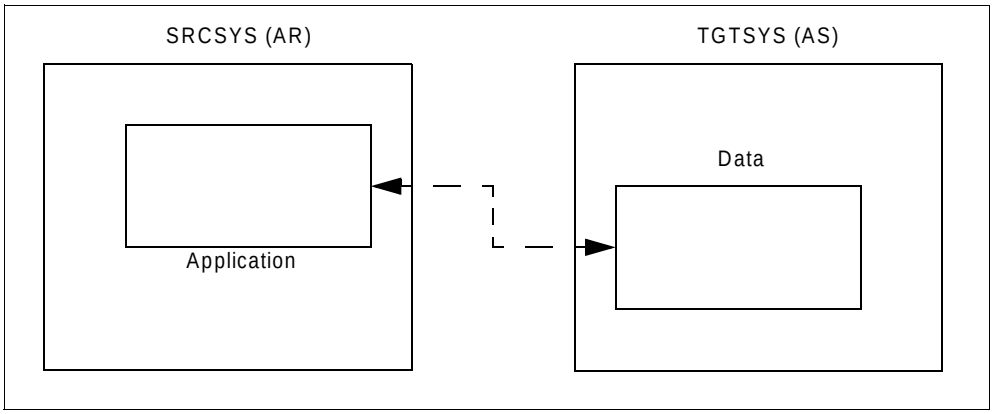


Figure 15. A Distributed Relational Database

To match our naming convention in the later chapters, the remote system TGTSYS takes the role of the AS and the source system SRCSYS, acting as an AR, and accesses the database located on system TGTSYS.

The configuration process for this simple scenario consists of two phases:

1. Setting up the Application Server (TGTSYS in our case). This phase involves the following configuration activities:
   a. Configure TCP/IP.
   b. Set the attributes for the DDM server job.

      Before you start the server job on the AS system, you can change the job's attributes with the Change DDM TCP/IP Attributes (CHGDDMTCPA) command. There are two attributes that can be changed with this command:

      - **AUTOSTART**—Specify whether to automatically start the DDM server when TCP/IP is started. The parameter takes effect the next time the Start TCP/IP (STRTCP) command is run.
      - **PWDRQD**—Specify whether client systems are required to have a password in addition to a user ID on incoming connection requests to this system as a server. This parameter takes effect on the next DRDA or DDM connect request over TCP/IP.

      The recommended values for the DDM server job are shown in Figure 16.

```
                Change DDM TCP/IP Attributes (CHGDDMTCPA)

Type choices, press Enter.

Autostart server . . . . . .   *YES          *NO, *YES, *SAME
Password required  . . . . . .   *YES          *NO, *YES, *SAME
```

Figure 16. DDM Server Job Attributes

   c. Start the DDM Server Job. Use the Start TCP/IP Server command to start the DDM server job:

```
STRTCPSVR SERVER(*DDM)
```

Figure 17. Starting the DDM Server Job

2. Setting up the Application Requester (SRCSYS). This phase involves the following configuration activities:
   a. Configure TCP/IP.
   b. Add the TGTSYS machine to the RDB directory. Use either WRKRDBDIRE or ADDRDBDIRE command.

```
                    Add RDB Directory Entry (ADDRDBDIRE)

Type choices, press Enter.

Relational database  . . . . . > TGTSYS        Character value
Remote location:
  Name or address  . . . . . . . > tgtsys.rochester.abc.com



  Type . . . . . . . . . . . . . > *IP           *SNA, *IP
Text . . . . . . . . . . . . . .   RDB Dir Entry for TGTSYS over TCP/IP

Port number or service program     *DRDA
```

Figure 18. RDB Directory Entry over TCP/IP

The description of the parameters for TCP/IP are the following:

- **Relational Database (RDB)**

  Same as for SNA.

- **Remote Location (RMTLOCNAME)**

  The first element of this parameter can take several forms:

  – IP address in dotted decimal form. Specify an Internet protocol address in the form nnn.nnn.nnn.nnn, where each nnn is a number in the range 0 through 255. If this form is used, the second element of this parameter must be specified as *IP.

  – IP host domain name. Specify an Internet host domain name of up to 254 characters in length. If this form is used, the second element of this parameter must be specified as *IP.

    If *IP is specified for the second element, the DRDA AS at the remote location must support the use of TCP/IP. The DEV, LCLLOCNAME, RMTNETID, MODE, and TNSPGM parameters will be ignored.

The optional parameters include:

- **Text (TEXT)**

  Same as for SNA.

- **Port number or service program (PORT)**

  Specify the TCP/IP port that is used at the remote location to communicate with the system on which the RDB is located. The default value of *DRDA is translated into port 446.

3. Defining the user profile under which to connect to TGTSYS. You can use the Add Server Authentication Entry (ADDSVRAUTE) command to add authentication information for a given user under which the connection is done. The user ID and password are associated with the user profile and remote Application Server. This information flows to the AS each time the AR issues a connect request.
   a. Before using the ADDSVRAUTE command, make sure that you have *SECADM special authority, as well as *OBJMGT and *USE authorities to the user profile to which the server authentication entry is added.
   b. Check whether the retain server security data (QRETSVRSEC) system value is set to 1. If the value is 0 (do not retain data), the password is not saved in the entry.
   c. Add the authentication entry with the ADDSVRAUTE command.

      Note: Please make sure that the server name parameter is in uppercase.

   The Change Server Authentication Entry (CHGSVRAUTE) command allows you to change the user ID and password for an authentication entry added by the ADDSVRAUTE command. The Remove Server Authentication Entry (RMVSVRAUTE) command allows you to remove authentication entries.

Now you can run the remote journal function over the TCP/IP protocol.

2.1.3 Preparing Local Journaling and Required Objects

The remote journal function is built upon the local journal function. Before the remote journal function can be performed, the local journal function on the source system must first be set up.

Create the local journal and the journal receiver on the source system by using the Create Journal Receiver (CRTJRNRCV) command and the Create Journal (CRTJRN) command respectively. Then, start local journaling on the source system using the Start Journal Physical File (STRJRNPF) command. Refer to *OS/400 Backup and Recovery*, SC41-5304, for more information on the local journal function.

Before you create your remote journal environment, make sure that you understand the performance benefits of the *RMVINTENT and *MINFIXLEN parameters on the CRTJRN command and the *OPNCLO parameter on the STRJRNPF command. Refer to 4.6, “Performance Recommendations” on page 79 for more details.

If library redirection is not specified, the remote journal and journal receivers will reside in a library that has the same name as the library that contains the source journal and journal receivers. You have to make sure that all selected libraries exist on the target system.

Figure 19 shows a remote journal created without library redirection. Journal entries in journal XMPJRN in library XMPLIB on the SRCSYS system are replicated to the remote journal that resides in the library of the same name on the TGTSYS system. The attached receiver RCVR0001 residing in library

RCVLIB in the SRCSYS system found the remote counterpart in the library of the same name on the TGTSYS system.

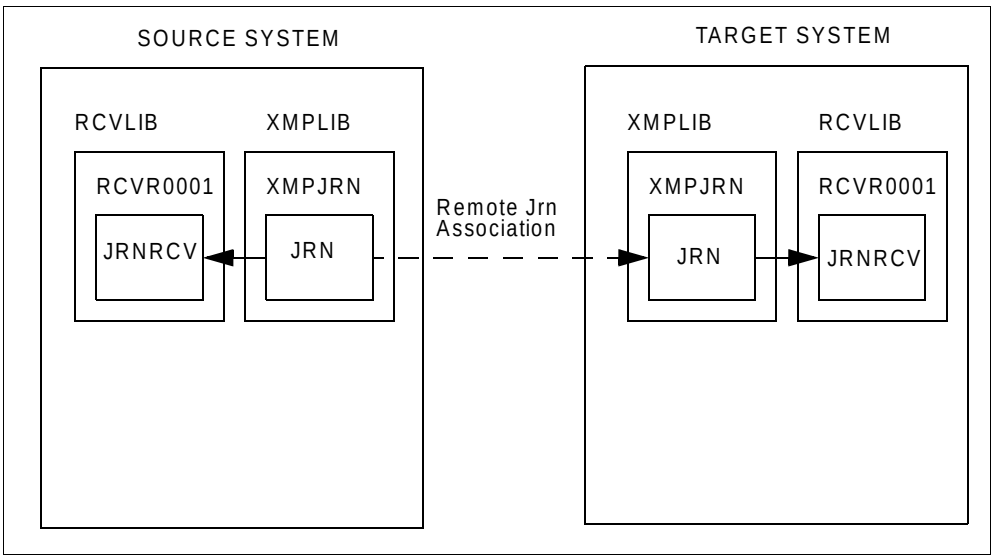


Figure 19. Remote Journaling without Library Redirection Specified

Using library redirection is another option you can choose. Library redirection basically 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. Except for the QGPL library, 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. The remote journal type specifies the extent of library redirection you can define. Please refer to 1.3.4, “Remote Journal Types” on page 8 for a more detailed discussion on library redirection.

Figure 20 on page 23 shows a remote journal created with library redirection. Journal entries from journal XMPJRN, in library XMPLIB on the SRCSYS system, are replicated to a remote journal in library XMPLIBBKP on the TGTSYS system. The attached receiver RCVR0001 residing in library RCVLIB in the SRCSYS system found the remote counterpart in the library RCVLIBBKP on the TGTSYS system.

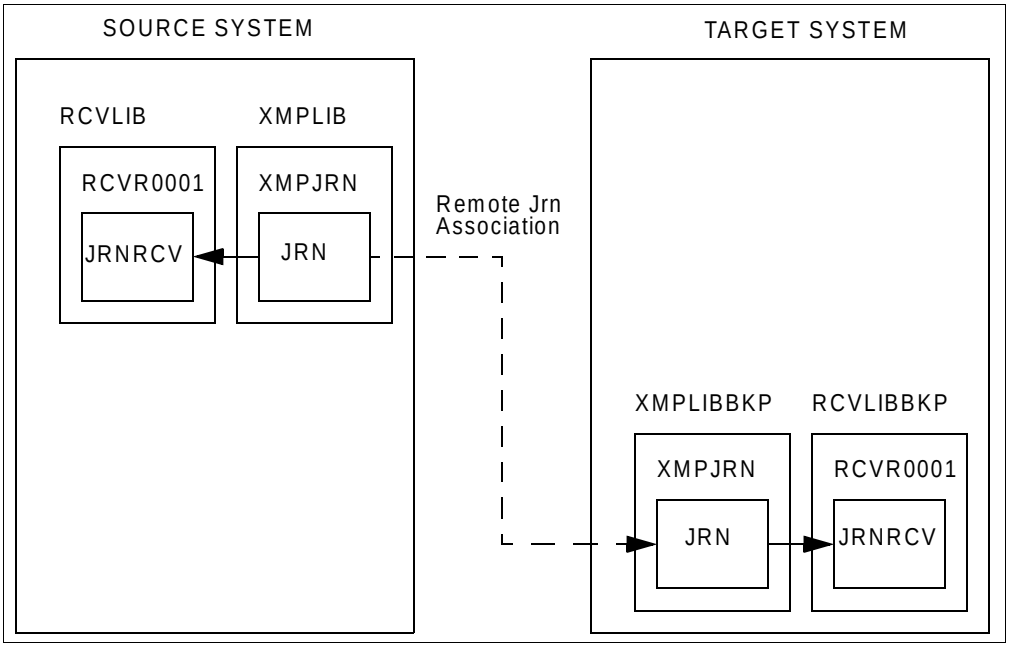


Figure 20. Remote Journaling with Library Redirection Specified

2.2 Adding Remote Journal (ADDRMTJRN)

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

In Figure 21 on page 24, a *TYPE1 remote journal named XMPJRN is created without library redirection on the TGTSYS system.

```
                        Add Remote Journal (ADDRMTJRN)

Type choices, press Enter.

Relational database  . . . . . .   TGTSYS
Source journal . . . . . . . . .   XMPJRN        Name
  Library  . . . . . . . . . . .     XMPLIB      Name, *LIBL, *CURLIB
Target journal . . . . . . . . .   *SRCJRN       Name, *SRCJRN
  Library  . . . . . . . . . . .                 Name
Remote receiver library  . . . .   *SRCRCVLIB    Name, *SRCRCVLIB
Remote journal type  . . . . . .   *TYPE1        *TYPE1, *TYPE2
Journal message queue  . . . . .   XMPJRNMSGQ    Name
  Library  . . . . . . . . . . .     XMPLIB      Name
Delete receivers . . . . . . . .   *NO           *NO, *YES
Text 'description' . . . . . . .   Remote Journal Example




                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 21. ADDRMTJRN Prompt Display

The following list further explains the parameters in the ADDRMTJRN command. Please note that *QJoAddRemoteJournal* API has similar input parameters.

- **Relational database (RDB)**

  Specify the name of the RDB directory entry that contains the remote location name of the target system. Refer to 2.1.2, “Setting Up a Relational Database Directory Entry” on page 14.

- **Source journal (SRCJRN)**

  Specify the name of the journal on the source system to which the remote journal is added and the library where it resides. The journal on the source system can be either a local journal or another remote journal. The name of the source journal can be qualified by specifying one of the valid library values.

- **Target journal (TGTJRN)**

  Specify the name of the remote journal on the target system.

- **Remote receiver library (RMTRCVLIB)**

  Specify the name of the library for the remote journal receivers on the target system that are associated with the remote journal.

- **Remote journal type (RMTJRNTYPE)**

  Specify the type of remote journal on the target system. The remote journal type influences the redirection capabilities, journal receiver restore operations, and remote journal association characteristics. Refer to 1.3.4, “Remote Journal Types” on page 8 for a description of the differences between the two available types.

- **Journal message queue (MSGQ)**

  Specify the name of the message queue associated with the remote journal. This value is only set for a journal that is created on the target system.

- **Delete receivers (DLTRCV)**

  Specify whether the system deletes the target journal receivers when they are no longer needed or keeps them on the target system for the user to delete after they are detached by the target system. This value is only set for a journal that is created on the target system.

- **Text "description" (TEXT)**

  Specify the text that briefly describes the remote journal on the target system. This value is only set for a journal that is created on the target system. Specify no more than 50 characters of text, enclosed in apostrophes.

When adding a remote journal to a source journal for the first time, the remote journal is created on the target system. The remote journal uses a combination of the attributes from the source journal and the input parameters provided on this command. The library in which the remote journal will be created must already exist on the target system prior to this command being called on the source system. When created by this command, the remote journal will be created with a journal type of *REMOTE. The remote journal will not have an attached journal receiver.

In addition, when adding the remote journal, the remote journal can either be created into the library with the same name as that of the source journal or into a redirected library on the target system. As described previously, the redirected library allows remote journals and any of their associated journal receivers to reside in libraries with different names on the target system from the corresponding local journal and journal receivers on the local system.

All validation for the journal library on the target system will be performed using the redirected library name when specified. Similarly, the journal receivers that are created later and associated with this remote journal can reside in the library with the same name as the source journal receivers on the source system. Or, they can reside in a distinct redirected library name on the target system. The journal receiver library redirection, if desired, must be specified when the remote journal is added using this command.

When adding a remote journal on a target system, two remote journal types can be specified: *TYPE1 and *TYPE2. The remote journal type influences the redirection capabilities, journal receiver restore operations, and remote journal association characteristics.

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

After the remote journal is successfully added on the target system, the remote journal has a journal state of *INACTIVE. A journal state of *INACTIVE for a remote journal on the target system means that the remote journal is currently not receiving journal entries from its source journal on the source system. The

Change Remote Journal (CHGRMTJRN) command or QjoChangeJournalState API is used to activate a remote journal and start the replication of journal entries from the source journal to the remote journal. Refer to Chapter 3, “Remote Journal Operations” on page 31, for the discussion of the CHGRMTJRN command and associated API.

The Work with Journal Attributes (WRKJRNA) command can be used to review journal attributes. Pressing F16 displays the journal state of the added remote journals. You can also activate and inactivate the remote journals on this panel using options 13 and 14. This is the same function that the CHGRMTJRN command and *QJoChangeJournalStatus* API provide.

Figure 22 shows that a remote journal XMPJRN is added using RDB entry TGTSYS. It is noted that the remote journal state is still *INACTIVE.

```
                    Work with Remote Journal Information

Journal  . . . . . :   XMPJRN          Library  . . . . . :   XMPLIB

Journal type . . . . :   *LOCAL        Journal state  . . . :   *ACTIVE
Remote journal type  :                 Delivery mode  . . . :
Local journal  . . . :                 Source journal . . . :
  Library  . . . . . :                   Library  . . . . . :
  System . . . . . . :                   System . . . . . . :
Redirected receiver library . . . . . . . . . . . . . . . . . :   XMPLIB
Number of remote journals . . . . . . . . . . . . . . . . . . :   1

Type options, press Enter.
  5=Display remote journal details   8=Display relational database detail
  13=Activate   14=Inactivate
                          ---------------------Remote----------------------
     Relational                                        Journal      Delivery
Opt  Database             Journal      Library        State        Mode
     TGTSYS               XMPJRN       XMPLIB         *INACTIVE
                                                                        Bottom
===>
F3=Exit       F4=Prompt   F5=Refresh   F6=Work with remote journal list
F9=Retrieve  F12=Cancel
```

Figure 22. WRKJRNA Display with a Remote Journal Added in a Remote System

Once a remote journal is added to a journal, the journal receiver that was attached at that time on the source system, and any journal receivers attached after that time on the source system, are protected from deletion. This occurs when all journal entries for a given journal receiver are not yet replicated to the remote journal. This protection ends when the remote journal is removed using the Remove Remote Journal (RMVRMTJRN) command or (QjoRemoveRemoteJournal) API. See 2.3, “Removing Remote Journal (RMVRMTJRN)” on page 28 for more information.

To setup a cascade configuration, you have to build an AS and AR relationship between the target system TGTSRC and a system other than the source system SRCSYS, for example, CASSYS. To achieve this relationship, you must set up the communications as described in 2.1.1, “Setting Up the Communications Environment and Security” on page 12. Then, you must establish the RDB directory entry as shown in 2.1.2, “Setting Up a Relational Database Directory Entry” on page 14, accordingly. Consideration of library creation and library

redirection is still applicable. Since the remote journal is created on the TGTSYS system, you can simply use ADDRMTJRN on TGTSYS to add a remote journal to the remote journal existing in the TGTSYS system to the CASSYS system. Journal entries replicated from SRCSYS to TGTSRC are, in turn, replicated to CASSYS.

Figure 23 illustrates the cascade configuration described previously in this section.

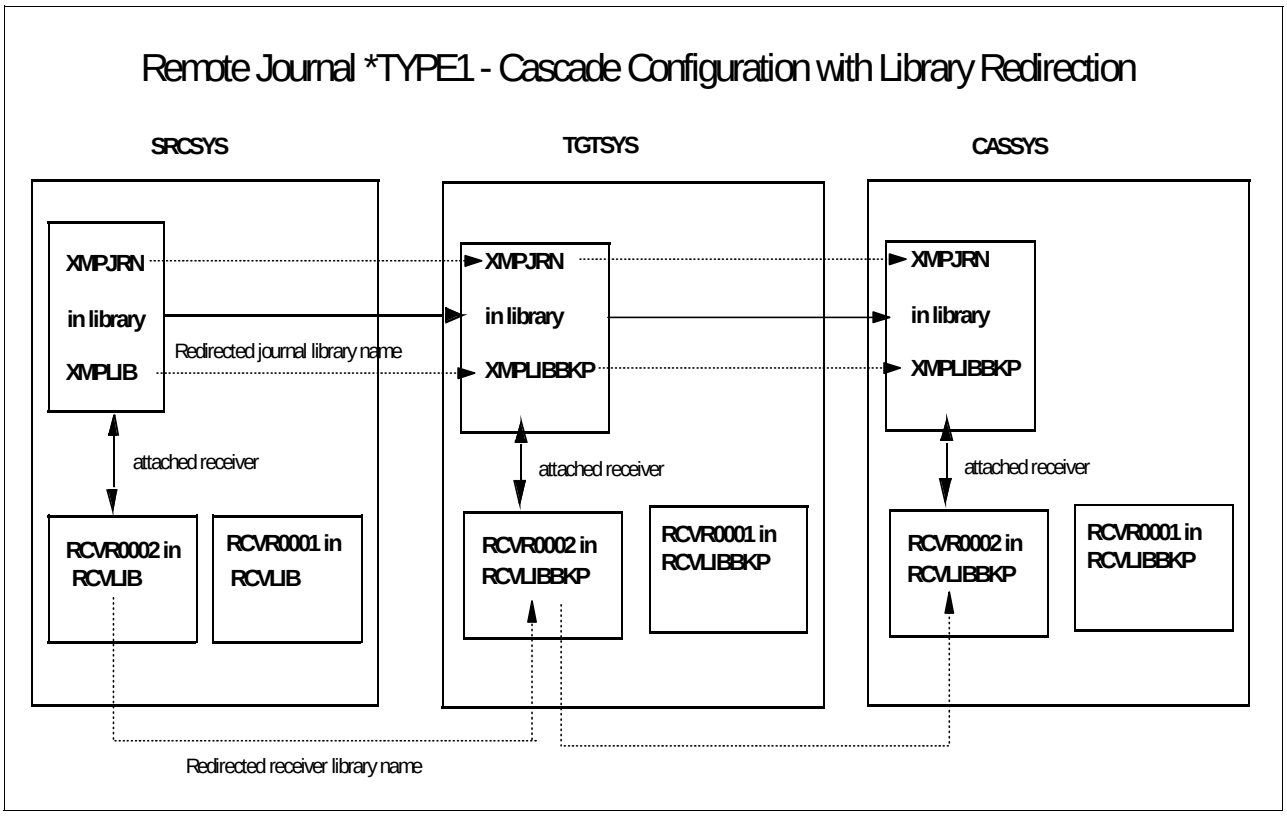


Figure 23. Sample Cascade Configuration

Figure 24 on page 28 shows a *TYPE1 remote journal on CASSYS added to the remote journal on the TGTSYS system in a cascade configuration. The command is entered on the TGTSYS system.

```
                    Add Remote Journal(ADDRMTJRN)

Type choices, press Enter.

Relational database  . . . . . .   CASSYS
Source journal . . . . . . . . .   XMPJRN        Name
  Library  . . . . . . . . . . .     XMPLIBBKP   Name, *LIBL, *CURLIB
Target journal . . . . . . . . .   *SRCJRN       Name, *SRCJRN
  Library  . . . . . . . . . . .                 Name
Remote receiver library  . . . .   *SRCRCVLIB    Name, *SRCRCVLIB
Remote journal type  . . . . . .   *TYPE1        *TYPE1, *TYPE2
Journal message queue  . . . . .   XMPJRNMSGQ    Name
  Library  . . . . . . . . . . .     XMPLIB      Name
Delete receivers . . . . . . . .   *NO           *NO, *YES
Text 'description' . . . . . . .   Cascade Remote Journal Example









                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 24. Adding a Remote Journal Cascade Configuration

In many AS/400 environments, users attempt to minimize the amount of processing that is performed on the local or source system. As much of the processing as possible is shifted to other AS/400 systems in the network. A combination of the broadcast and cascade configurations allows for this when replicating the journal entries from a single AS/400 system to multiple AS/400 systems. For example, replicating a local journal to a single remote journal on a target AS/400 system minimizes the replication cost on the source 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 AS/400 systems.

2.3 Removing Remote Journal (RMVRMTJRN)

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

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

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

Once a remote journal association is removed from a journal, all of the journal receivers that are currently in the journal's receiver directory on the source

system are no longer protected from deletion even if the journal entries are not yet replicated to the remote journal.

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

Figure 25 shows a prompt display of the RMVRMTJRN command, which is used to remove remote journal XMPJRN from a source journal of the same name.

```
                    Remove Remote Journal (RMVRMTJRN)

Type choices, press Enter.

Relational database  . . . . . .   TGTSYS
Source journal . . . . . . . . .   XMPJRN        Name
  Library  . . . . . . . . . . .     XMPLIB      Name, *LIBL, *CURLIB
Target journal . . . . . . . . .   *SRCJRN       Name, *SRCJRN
  Library  . . . . . . . . . . .                 Name

                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 25. RMVRMTJRN Prompt Display

The following list further explains the parameters on the RMVRMTJRN command. The *QjoRemoveRemoteJournal API* has similar input parameters.

- **Relational database (RDB)**

  The name of the RDB directory entry that contains the remote location name of the target system. Refer to 2.1.2, “Setting Up a Relational Database Directory Entry” on page 14 to create an RDB directory entry.

- **Source journal (SRCJRN)**

  Specify the name of the journal on the source system from which the remote journal is removed, and the library where it resides. The journal on the source system can be either a local journal or a remote journal.

- **Target journal (TGTJRN)**

  Specify the name of the remote journal on the target system.

Before you use RMVRMTJRN to remove the association between the journal in source and target systems, you must make sure that the journal state of the remote journal is inactive. Otherwise, message CPF6992 will appear saying that the remote journal in the library was not removed because the journal state of the remote journal is *ACTIVE. You must change the journal state of the remote

journal by using the Change Remote Journal (CHGRMTJRN) command, which is described in 3.2, “De-activating a Remote Journal” on page 43.

2.4 Version Considerations

AS/400 systems that use the remote journal function must be at V4R2M0 or above. Only receivers attached to a journal after installing V4R2M0 can be replicated by the remote journal function. All journals are regarded as local journals if they were created prior to V4R2M0. There is no means for converting existing journal receivers from releases prior to V4R2M0 to a form that allows the remote journal function.

As a step in the upgrade process from any prior version to V4R2M0 or above a Change Journal (CHGJRN) command was run for all journals on the system to attach newly created journal receivers of the upgrade version. After this step, these journals can handle the remote journal function.

Chapter 3. Remote Journal Operations

This chapter addresses the operations of a remote journal. Operational tasks that are discussed are:

- Activating a remote journal
- Deactivating a remote journal
- Determining the status of a remote journal
- Managing journal receivers on both the source and target systems
- Managing multiple remote journals
- Monitoring for errors
- Recovery from communication failures
- Save and restore considerations
- High availability considerations

The operation of remote journal is discussed and illustrated by the use of CL commands. Each remote journal CL command also has an equivalent API that can be used instead. Table 1 shows the correspondence between each remote journal CL command and the equivalent remote journal API. The APIs were available in V4R2M0, while the commands were available in V4R3M0.

Table 1. Command to API Correspondence

Command	Callable API
ADDRMTJRN (Add Remote Journal)	QjoAddRemoteJournal
CHGRMTJRN (Change Remote Journal)	QjoChangeJournalState
RMVRMTJRN (Remove Remote Journal)	QjoRemoveRemoteJournal

3.1 Activating a Remote Journal

Before journal entries can be replicated to a remote journal, that remote journal must be activated. To activate any remote journal, use the Change Remote Journal (CHGRMTJRN) command. This command must be run on the *source* system, on which the source journal is located. Once the remote journal is activated, the OS/400 system support starts replicating journal entries from the source journal to that particular remote journal. Each remote journal must be activated individually by using this command. Activating one remote journal does not impact the current state of any other remote journal including those that are associated with the same source journal as the one being activated.

3.1.1 Using the CHGRMTJRN Command for Activation

When preparing to run the CHGRMTJRN command to activate a remote journal, choices must be made for the following parameters:

- **Relational Database (RDB)**

  This parameter specifies the relational database name. The name specified here identifies an entry in the relational database (RDB) directory which was previously added by using the ADDRDBDIRE command. The RDB directory entry selected provides the following information to the remote journal system support:

– Identification of the target system
– The type of communication path to use when communicating with that target system
– The type of communications protocol to use (SNA, OptiConnect, or TCP/IP).

It is possible to have more than one directory entry for the same target system, where each one specifies a different communications path and protocol. For example, one entry can specify that OptiConnect/400 with SNA is to be used for communications with the target system and a second can specify that ATM with TCP/IP is to be used. The choice that is specified on the CHGRMTJRN command determines which communication path replicated journal entries will be transported across. The choice that is made on the command stays in effect until the remote journal is first deactivated and then activated again using a different choice. The choices that are available can be viewed by running the WRKRDBDIRE command. However, the choice you use on the CHGRMTJRN command must be the same as the choice you used when you ran the ADDRMTJRN command.

Attention

You can run multiple ADDRMTJRN commands that specify the same source and remote journals and only vary by having different RDB values specified. When you do this, you prepare alternative communication paths between the same source journal and remote journal that can be used for journal entry replication. Only one of these paths can be active at a time. If you activated a remote journal using one path and want to change it, you must first deactivate the remote journal and then activate it again using the new RDB directory entry. Using this technique, you can prepare for recovery from a communications failure. This is discussed later in 3.7, “Recovering from Communication Failure” on page 62.

- **Source Journal (SRCJRN)**

  This parameter specifies the qualified name of the source journal. Journal entries in the receiver chain of this journal are replicated to the remote journal that is being activated. The journal that is identified here must exist on the source system at the time the CHGRMTJRN command is run. The source journal can be a *local* journal or a *remote* journal.

  At the time the command is run, there may already be journal entries in the receiver chain that are eligible for replication to the remote journal. If so, the system support immediately starts transporting journal entries to the remote journal as part of running this command. This is referred to as *catch-up mode*. Conversely, at the time the command is run, there may be no eligible journal entries in the receiver chain. In that case, no journal entries are transported to the remote journal as part of running the command. The remote journal is simply put into a state that when the first eligible journal entry arrives on the source journal it can be replicated to the remote journal. This is the normal operational mode once the remote journal is activated and is referred to as *continuous mode*.

  Observe caution when using the CHGRMTJRN command interactively. The source journal may have a large number of pending journal entries that must be replicated to the remote journal that is being activated. In this case, the

command will not complete until those journal entries are replicated. The keyboard is disabled during this time period so no additional commands can be entered. If there is any question as to whether running the command results in a prolonged catch-up phase, the command should be submitted as a batch job.

- **Target Journal (TGTJRN)**

  This parameter specifies the remote journal on the target system to which journal entries are to be replicated from the source journal. This remote journal must have been added previously to the source journal by using the ADDRMTJRN command.

- **Journal State (JRNSTATE)**

  This parameter specifies how the state of the remote journal is to be changed. To activate a remote journal, use the key value *ACTIVE.

- **Delivery (DELIVERY)**

  This parameter specifies whether replicated journal entries are to be sent to the remote journal synchronously to the application that generated them or asynchronously to that application. If the key value *SYNC is specified, the entries are sent synchronously to the application. If the key value *ASYNC is specified, they are sent asynchronously to the application.

Attention

There is an important relationship between this parameter and the RDB parameter described in the previous text. If *SYNC is specified for this parameter, ensure that the entry specified in the RDB parameter identifies a properly-sized communications gear. For performance information, refer to Chapter 4, “Remote Journal Performance” on page 69.

- **Starting Journal Receiver (STRJRNRCV)**

  This parameter specifies where, in the receiver chain of the source journal, replication is to start. There are three possible choices for this parameter. Each of these three choices serves a different purpose and is used under differing circumstances.

Attention

The choice of the starting point must be made carefully because the wrong choice can cause journal entries to be missed. Moreover, recovery of missed journal entries can be difficult. A more extensive discussion of choosing the starting point for remote journal is given in 3.1.3, “Choosing a Starting Point for Remote Journal” on page 36.

- **Sending Task Priority (SNDTSKPTY)**

  This parameter is used only when the DELIVERY parameter specifies *ASYNC. The value specified here controls the priority at which the system sending task, which replicates journal entries to the remote journal, runs. The lower the numeric number is that is specified here, the higher the priority of this system task is. Depending on the load on the source system, a high priority may be needed to help ensure that the remote journal does not lag behind the source journal by a significant number of entries. Allowing the

remote journal to significantly lag behind the source journal must be avoided for an effective high availability solution. Considerations for high availability are discussed later in Chapter 5, “Building the Switchover Solution” on page 81.

Note

The default value *SYSDFT is the highest priority that the independent sending task can have. If you enter a number (from 1 to 99), the result is a lower priority than the default.

3.1.2 Activating the Remote Journal Using CHGRMTJRN—Example

In this section, we present a simple example of activating a remote journal. The system configuration we work with is illustrated in Figure 26. The source system, SRCSYS, contains the source journal XMPJRN in library XMPLIB. This journal is also a local journal into which applications cause journal entries to be deposited. As shown, there are two receivers in the receiver chain of XMPJRN: RCVR0001 and RCVR0002. The receivers are in the RCVLIB library. The target system TGTSYS contains the remote journal XMPJRN, in library XMPLIBBKP on the target system. Previously the ADDRMTJRN command was run to add this *TYPE1 remote journal to the source journal. This is shown by the dashed line between the two. At this point in time, the remote journal is not yet activated, and there are no journal receivers in its receiver chain. When the remote journal is activated, journal receivers start to be created on the target system as journal entries are replicated. In this example, we assume that the library for the journal receivers on the target system was defined on the ADDRMTJRN command to be RCVLIBBKP.

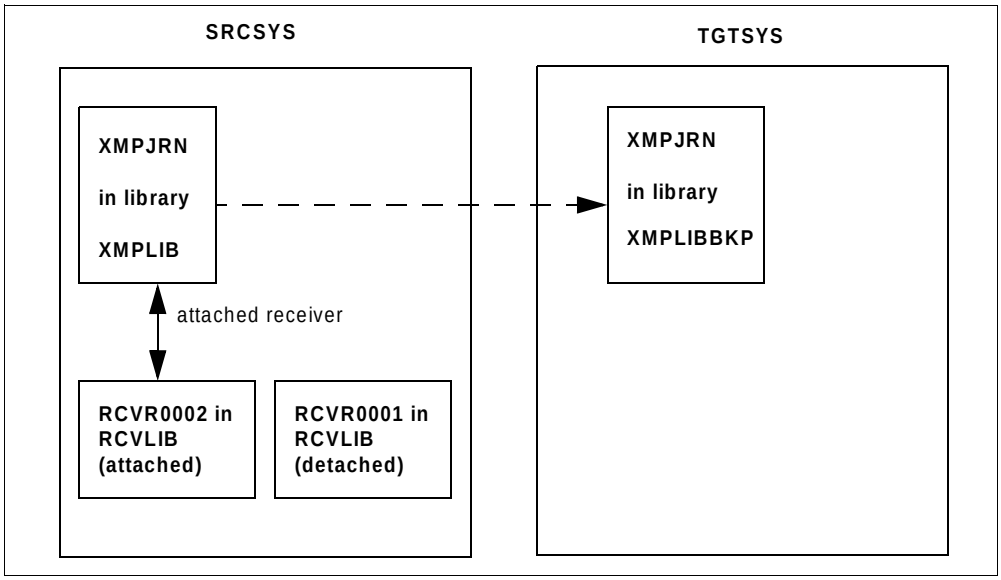


Figure 26. Remote Journal Configuration before Running the CHGRMTJRN Command

To activate the remote journal XPMJRN on the target system, we use the CHGRMTJRN command. The prompt screen (with the parameter values filled in) for this command is shown in Figure 27 on page 35.

```
                    Change Remote Journal (CHGRMTJRN)

Type choices, press Enter.

Relational database  . . . . . > TGTSYS
Source journal . . . . . . . . > XMPJRN        Name
  Library  . . . . . . . . . . >   XMPLIB      Name, *LIBL, *CURLIB
Target journal . . . . . . . .   XMPJRN        Name, *SRCJRN
  Library  . . . . . . . . . .     XMPLIBBKP   Name
Journal state  . . . . . . . . > *ACTIVE       *SAME, *ACTIVE, *INACTIVE
Delivery . . . . . . . . . . .   *ASYNC        *ASYNC, *SYNC
Starting journal receiver  . . > RCVR0001      Name, *ATTACHED, *SRCSYS
  Library  . . . . . . . . . . >   RCVLIB      Name, *LIBL, *CURLIB
Sending task priority  . . . . .   *SYSDFT      Number, *SYSDFT

                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 27. CHGRMTJRN Prompt Display—Activate Remote Journal

From this figure, you can see that the remote journal is activated using the asynchronous delivery mode with the system delivery task to run with a priority of *SYSDFT. We recommend that you do not to change the default priority. The RDB directory entry is specified as the same name as the target system. It is good practice to name the entry the same as the target system to avoid confusion. The directory entry itself does not directly identify the target system. By using a naming convention such as this, it is readily understood which directory entry is for which system.

Notice also that the starting point for journal entry replication is specified as RCVR0001, which is the first receiver on the source journal's receiver chain. We want to replicate all journal entries. Since journal replication only moves forward in the receiver chain, by specifying the first receiver, we are assured that we will not miss any journal entries. If we specified the key value *SRCSYS or *ATTACHED, we could have started with the receiver that is currently attached to the source journal (RCVR0002). The journal entries in receiver RCVR0001 would not be replicated.

You cannot provide a journal entry sequence number when specifying the starting point. When remote journaling is activated, the system support on each side exchange information to determine which journal entries are already replicated to the target side. From this, the system support on the source side can automatically determine at which journal entry to start. In our example, since no journal entry replication done yet, the system knows to start with the first journal entry in the specified receiver. For example, we may allow journal entries to be replicated for a while and then deactivate the remote journal. The next time we run the CHGRMTJRN command, while still specifying RCVR0001 as the starting point, the system support automatically determines which entries in this receiver

are already replicated. It starts with the first one that was not replicated. Journal entries that are already replicated are not replicated a second time.

If we press Enter on the illustrated prompt display and activate the remote journal, when the command completes, the system configuration in Figure 26 on page 34 transforms to the configuration in Figure 28. Notice that all pending journal entries on the source system are replicated to the target system and that the remote journal now has a receiver chain. This catch-up phase occurred synchronous to the running of the CHGRMTJRN command. Now as applications on the source system cause entries to be deposited in the source journal, the entries are replicated to the remote journal by the asynchronous delivery system task.

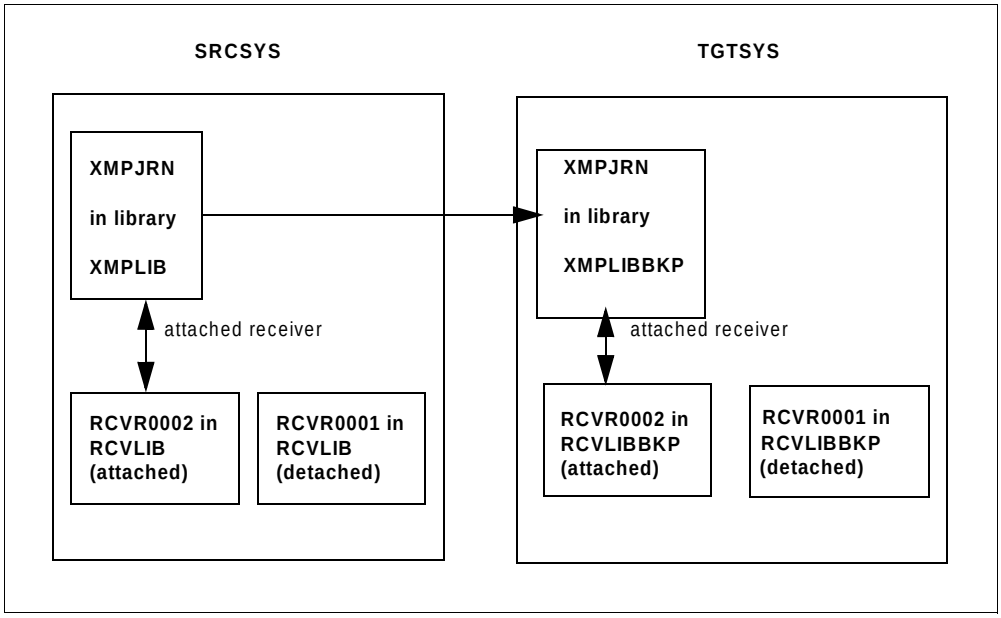


Figure 28. Simple Remote Journal Configuration after Running CHGRMTJRN Command

3.1.3 Choosing a Starting Point for Remote Journal

When activating a remote journal through the CHGRMTJRN command, you must identify where the replication of journal entries is to start. This identification provides a starting point within the receiver chain of the *source* journal. You cannot identify a single journal entry. You can only identify a receiver in the source journal's receiver chain. Given the receiver, the system support determines which, if any, of the journal entries in that receiver have already been replicated. The system support searches forward through the receiver until it finds the first entry that is not yet replicated. Journal entry replication starts with that journal entry. If the system support does not find any entries that are not yet replicated, the next receiver in the chain will be searched. This is repeated until an entry is found in a receiver. If there are no journal entries in any of the receivers on the chain that need to be replicated, the system support awaits the arrival of the first new journal entry. Then, replication commences with that one.

One important point to note is that the system support always moves forward in the receiver chain, from oldest to newest journal entry. It starts at the point that you identify on the CHGRMTJRN command. This means that journal entries are

replicated in the same order as they were deposited into the source journal. This, in turn, maintains the correct ordering of the journal entries as they are deposited into the remote journal. Maintaining this correct order in the remote journal is critical to having a high availability solution. A high availability solution works because it replays changes made to the source database onto a backup database that is on the target system. To ensure that the source database and the backup remain at the same level, the changes represented by the journal entries must be applied to the backup database in the same order as the changes that were originally made to the source database. If the changes represented by the journal entries are applied in a different order, the backup database does not even closely reflect the current level of the source database.

By incorrectly choosing the starting point, you can inadvertently change the order of the journal entries in the remote journal. For example, if you direct the support to start in the middle of the receiver chain, journal entries older than those at the specified starting point are not looked at. If these entries were not replicated previously, they are not replicated this time either. There are legitimate reasons for skipping journal entries. However, if this is done by mistake, replicating the missed journal entries later may cause the journal entries to be replayed by the apply jobs of the high availability application in a wrong order. For example, study Figure 29.

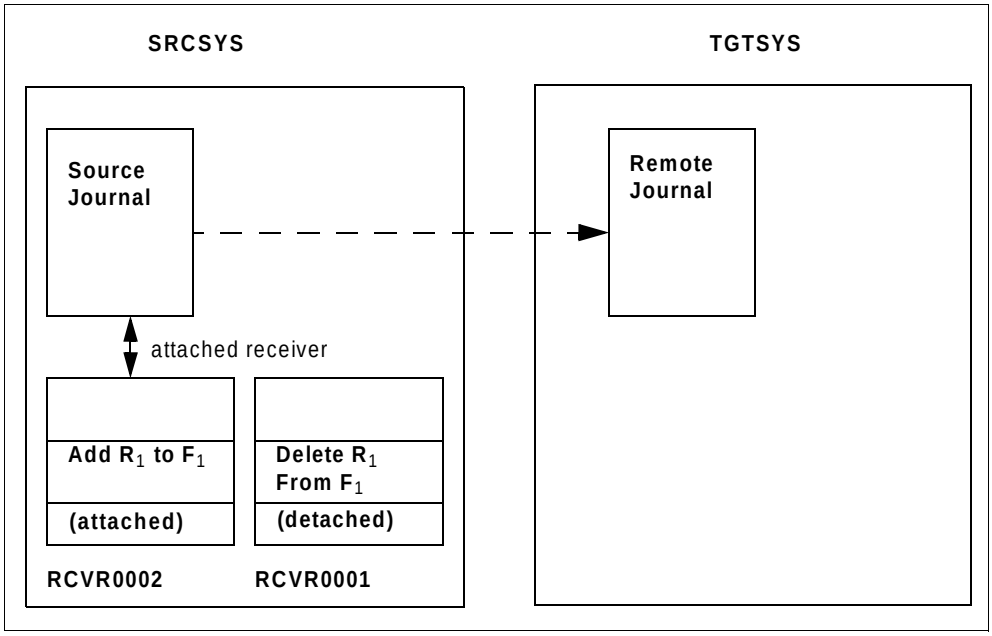


Figure 29. Getting Journal Entries Out of Sequence (Part 1 of 3)

In this example, the source journal has two receivers in its receiver chain: RCVR0001 and RCVR0002. Receiver RCVR0002 is currently attached to the journal. Therefore, the journal entries in this receiver are newer than the journal entries in RCVR0001. Also note that receiver RCVR0001 contains a journal entry that represents deleting record R_1 from physical file F_1. In RCVR0002, there is an entry that represents the insertion of record R_1 into physical file F_1. Record R_1 was first deleted from the file and then re-inserted later.

Suppose we use the CHGRMTJRN command to activate the remote journal and specify the receiver RCVR0002 as the starting point. Upon completing the command, we have the configuration shown in Figure 30. Note that receiver RCVR0002 is replicated, but not RCVR0001 because our specified starting point precluded replication of this receiver.

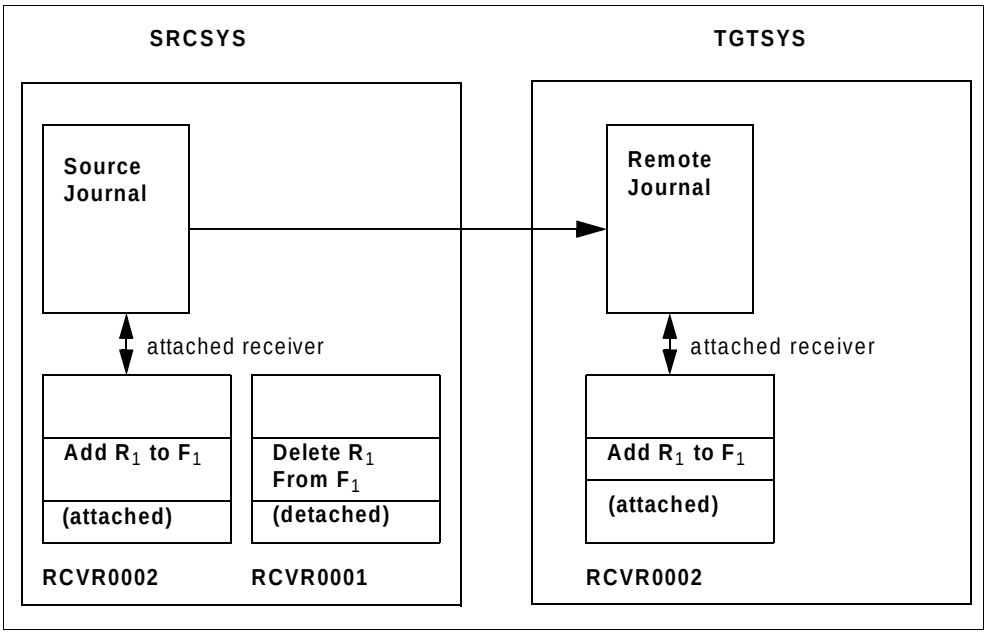


Figure 30. Getting Journal Entries Out of Sequence (Part 2 of 3)

Suppose we recognize our mistake. Then, we deactivate the remote journal and then reactivate, this time specifying RCVR0001 as the starting point. Two actions now happen. Receiver RCVR0001 is replicated to the remote journal but it appears as a newer receiver than RCVR0002. The receivers on the chain of the remote journal could temporarily appear to apply jobs of a high availability solution as being out of order as shown in Figure 31 on page 39

Note: The remote journal moves itself correctly in order after the next CHGJRN command execution.

If journal entries are applied to the backup database in receiver order, we perform an add first of record R1 followed by a delete, which is the opposite order as on the source database. The second action that happens is that the system support ends with an error condition. Once receiver RCVR0001 is replicated, the system support tries to replicate RCVR0002 again. It realizes that this receiver is already replicated and the receivers are probably out of order on the remote journal. The remote journal is automatically deactivated so replication cannot continue.

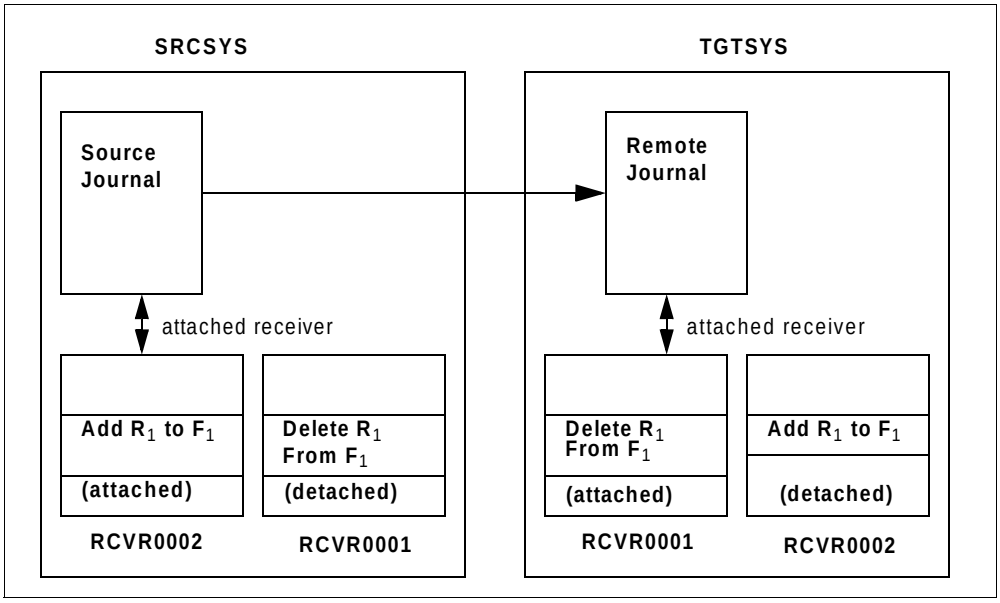


Figure 31. Getting Journal Entries Out of Sequence (Part 3 of 3)

The CHGRMTJRN command supports three ways of specifying the starting point when activating a remote journal:

- **Use the key value *ATTACHED.**

  *ATTACHED means that replication must start with the receiver that is currently attached to the remote journal. Based on what the attached receiver is on the remote journal, the system finds the corresponding receiver in the receiver chain of the source journal and replication of journal entries proceed from that point. If there is currently no receiver attached to the remote journal, replication starts with the receiver that is currently attached to the source journal.

  This choice is used when the remote journal was previously activated and later deactivated. During the time period where the remote journal is deactivated, additional journal entries are deposited into the source journal. Now you want to continue replication to the remote journal starting at the next journal entry that is not yet replicated. Using this choice ensures that all journal entries deposited in the source journal are replicated, in the correct order, to the remote journal without having the remote journal active all the time. Use caution when activating a remote journal for the first time. If a remote journal is not yet activated, it does not have an attached receiver. Therefore, if the starting point is specified as *ATTACHED, replication starts with the receiver that is attached to the source journal. If the receiver chain of the source journal only consists of the currently attached receiver, no journal entries are skipped by the replication process. If there are more receivers than the currently attached one, journal entries in receivers other than the currently attached one are not replicated.

  Using the *ATTACHED key value is illustrated in Figure 32 on page 40.

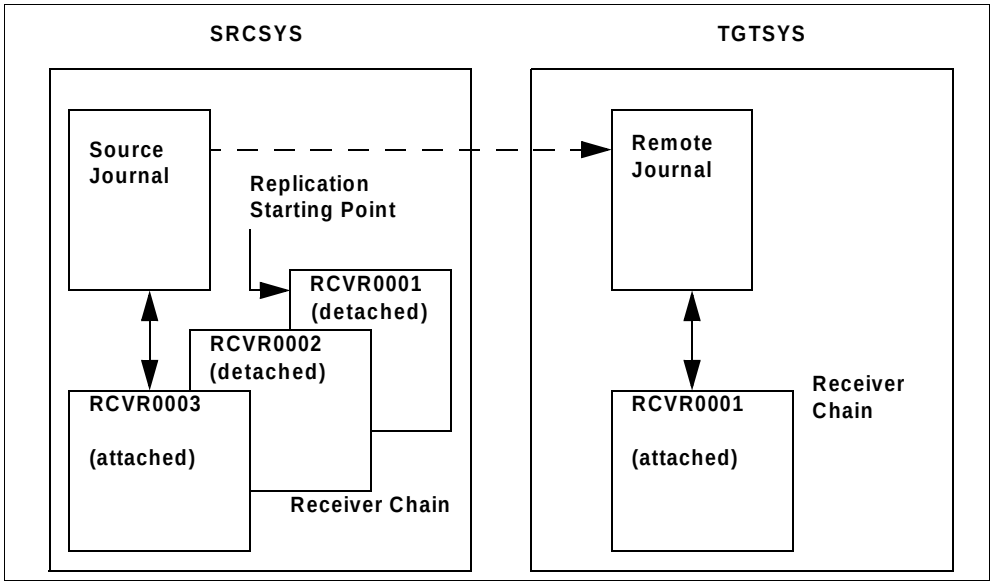


*Figure 32. Example of Using *ATTACHED*

In this example, the remote journal has been activated once before and journal entries in receiver RCVR0001 were replicated before the remote journal was deactivated. Receiver RCVR0001 remains attached to the remote journal, and there are no other receivers on the chain. While the remote journal was inactive, additional entries could have been deposited into RCVR0001 and receivers RCVR0002 and RCVR0003 were added to the receiver chain of the source journal. RCVR0003 is the receiver currently attached to the source journal. If we activate the remote journal again, specifying *ATTACHED as the starting point, the system support determines if RCVR0001 is currently attached to the remote journal. It finds that same receiver on the chain of the source 1 journal. Starting with that receiver, the system searches forward through the journal entries until it finds the first entry that was not replicated to the remote journal. Replication starts at that point and proceed forward through the receivers on the source journal chain. As a result, all journal entries deposited into the source journal receivers are replicated to the remote journal in the correct order.

- **Use the key value *SRCSYS.**

  *SRCSYS means that replication must start at the receiver that is currently attached to the source journal. If there are other receivers in the receiver chain of the source journal, the journal entries they contain are not replicated.

  When *SRCSYS is specified, the system support checks the remote journal to determine if the receiver attached to the source journal is already somewhere on the receiver chain of the remote journal. One of three possible outcomes result:

  – The receiver is not on the receiver chain of the remote journal. In this case, a new receiver with this name is created and attached to the remote journal. Replication of journal entries proceeds, starting with the first journal entry in that receiver on the source system.

  – The receiver is already on the receiver chain of the remote journal and is currently attached to the remote journal. In this case, a portion of the

journal entries in the receiver may already be replicated. The system support determines the first journal entry in the receiver that is not yet replicated. Replication starts from that point.

– The receiver is already on the receiver chain of the remote journal and is not the one that is currently attached to the remote journal. In this case, the remote journal cannot be activated, and the CHGRMTJRN command fails. Once a receiver is replicated to the remote journal, it cannot be replicated a second time after it is detached from the remote journal. You can delete the receiver from the target system and replicate it a second time. However, this may result in journal receivers that are out of order in the remote journals receiver directory until the receiver is detached from the remote journal. At that point, it is placed into the correct location in the chain.

The key value *SRCSYS is useful in two situations:

– The only receiver on the chain of the source journal is the currently attached receiver, or all other receivers are already replicated to the remote journal. Used in other situations, you must take care so journal entries are not inadvertently skipped by the replication process.

– It becomes necessary to prevent journal entries from being replicated. For example, an application, due to an error, generated erroneous transactions that should not be repeated on the target system. By running the CHGJRN command to attach a new journal receiver and activating the remote journal using the *SRCSYS key value, the starting point for replication can be moved beyond the journal entries that are not to be replicated.

An example of using *SRCSYS is shown in Figure 33. In this example, we deliberately prevent journal entries in receivers RCVR0002 and RCVR0003 from being replicated.

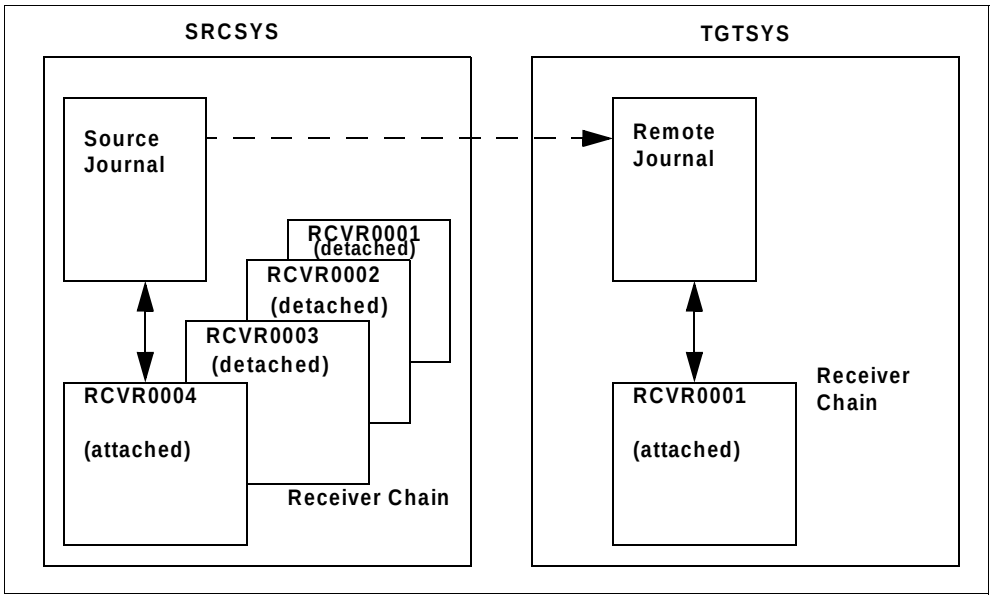


*Figure 33. Example of Using *SRCSYS before Activation*

The remote journal was activated at a previous time to allow RCVR0001 to be replicated. Then it was deactivated. While it was in an inactive state, receivers RCVR0002 to RCVR0004 were added to the receiver chain of the source journal. Now we want to activate the remote journal again using *SRCSYS as the starting

point for replication. This causes the system support to start with receiver RCVR0004. The end result of using *SRCSYS in this example is shown in Figure 34.

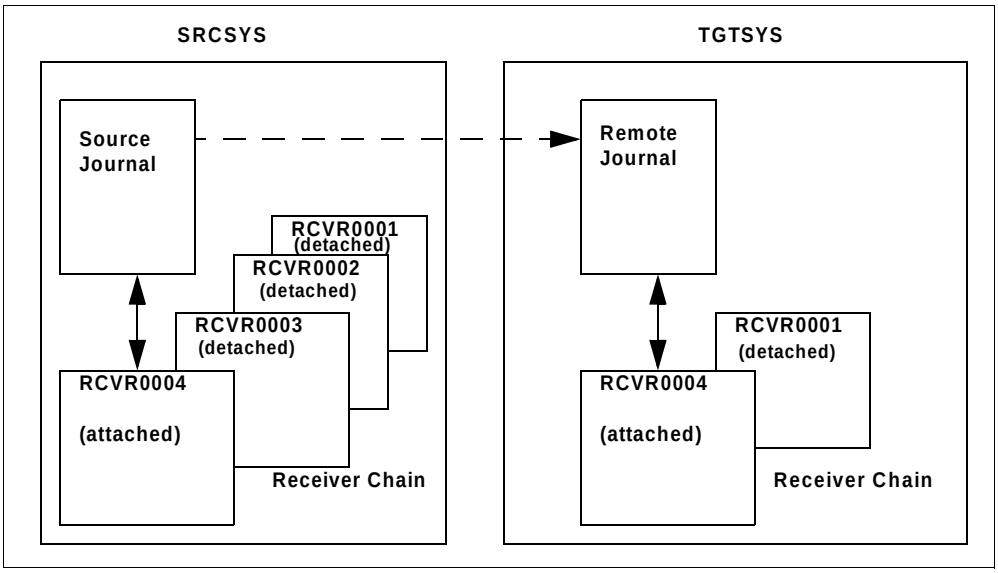


*Figure 34. Example of Using *SRCSYS after Activation*

- **Use the receiver Name.**

  The starting point can be given explicitly by specifying the name of a receiver that is associated with the source journal. When this option is used, the system support checks if the named receiver is already on the receiver chain for the remote journal. Again, one of three outcomes is possible when this check is made:

  – The receiver is not on the receiver chain of the remote journal. In this case, a new receiver with this name is attached to the remote journal and replication of journal entries proceeds, starting with the first journal entry in that receiver on the source system.

  – The receiver is already on the receiver chain of the remote journal and is currently attached to the remote journal. In this case, a portion of the journal entries in the receiver may already be replicated. The system support determines the first journal entry in the receiver that is not yet replicated. Replication starts from that point.

  – The receiver is already on the receiver chain of the remote journal and is not the one that is currently attached to the remote journal. In this case, the remote journal cannot be activated, and the CHGRMTJRN command fails. Once a receiver is replicated to the remote journal, it cannot be replicated a second time after it is detached from the remote journal. You can delete the receiver from the target system and replicate it a second time. However, this may result in journal receivers that are out of order in the remote journals receiver directory until the receiver is detached from the remote journal at which time it is placed into the correct location in the chain.

  Specifying a receiver name for the starting point is useful in two situations:

  – Activating a remote journal for the first time where the source journal has more receivers on its chain than the currently attached one. To

replicate all journal entries in this case, you specify the name of the oldest receiver on the receiver chain.

– Wanting to prevent the journal entries in some receivers from being replicated. By specifying the name of a newer receiver in the receiver chain journal entries in the older receivers are precluded from being replicated.

Previous examples (Figure 26 on page 34 and Figure 28 on page 36) illustrate the use of a receiver name for specifying a starting point.

3.1.4 Ineligible Journal Receivers

Receivers that were created prior to V4R2 are not eligible for use with the remote journal support. At the time, a system is upgraded from a release prior to V4R2, each journal that exists on the system is changed by forcing a new receiver to be attached to it. Any existing receivers are left on the receiver chain of the journal. If you attempt to use the CHGRMTJRN command, specifying the name of these old receivers as the starting point, the command fails. The remote journal is not activated.

Unfortunately there is no attribute associated with a receiver to determine if the receiver can be used with the remote journal support. The only way to determine if a receiver is eligible is to examine the date it was detached from the journal. If that date is the same as or before the date the system was upgraded, that receiver and all receivers older than that one are ineligible for use with remote journal support.

If you have ineligible receivers on the receiver chain, the first time you activate a remote journal, you can skip over them by specifying a starting point by receiver name. Specify the name of a receiver that is newer on the chain than the ones that are ineligible. Because the remote journal only moves forward in the receiver chain, the support will never encounter any further ineligible receivers. Please note also that dual receivers are not allowed in the receiver chain.

3.2 De-activating a Remote Journal

To end the replication of journal entries to a remote journal, the remote journal must be deactivated. Once deactivated, the system support no longer replicates journal entries to that remote journal. New entries that arrive in the source journal are only deposited in the local journal receiver until the remote journal is activated again.

Each remote journal must be deactivated individually. Deactivating one remote journal does not change the state of any other remote journal, even those that are connected to the same source journal. To deactivate a remote journal, one of two commands can be used:

- The CHGRMTJRN command can be used to deactivate a remote journal from the *source* system.
- The Change Journal (CHGJRN) command can be used to deactivate a remote journal from the *target* system.

Each of these two commands are discussed in the following sections in more detail.

3.2.1 Using the CHGRMTJRN Command for Deactivation

When preparing to run the CHGRMTJRN command to deactivate a remote journal, choices must be made for the following parameters:

- **Relational Database (RDB)**

  Specify an RDB directory entry that identifies the communications path to use to the system on which the remote journal exists. You must specify the same RDB entry that was used to activate the remote journal. In the event you have multiple RDB entries to that target system, you are not allowed to use a different one.

- **Source Journal (SRCJRN)**

  This parameter specifies the qualified name of the source journal from which entries are currently replicated to the remote journal.

- **Target Journal (TGTJRN)**

  This parameter specifies the name of the remote journal to be deactivated.

- **Journal State (JRNSTATE)**

  This parameter specifies how the state of the remote journal must be changed. To deactivate a remote journal, the key value *INACTIVE is used.

- **How to make inactive (INACTOPT)**

  This parameter specifies how deactivation occurs. One of two key values can be specified:

  – ***CNTRLD**

    This option allows those journal entries that are queued up for replication to complete. At any point, there may be one or more journal entries that are prepared for transmission to the remote journal. Using this option allows the transmission of those entries to complete before the remote journal is deactivated. Note that if the asynchronous delivery mode is used over a slow communications line, there can be many journal entries queued up. As a result, it can be a significant time period before the remote journal is actually deactivated. This value has no meaning for remote journals activated in synchronous mode.

  – ***IMMED**

    This option is used to deactivate the remote journal immediately. Journal entries that are already queued up for transmission are not sent before the remote journal is deactivated. Note that this does not cause the journal entries to be lost to that remote journal. The next time the remote journal is activated, those same journal entries are prepared again for transmission to the remote journal.

Note: The *QjoChangeRemoteJournal* API differs from the CHGRMTJRN command in the area of deactivation. Unlike the command, the API can be used to deactivate a remote journal from either the source or target side. In addition, information is returned to the API caller with details about the inactivate request such as the sequence number of the last journal entry that was replicated.

To deactivate the remote journal, which was activated in Figure 27 on page 35, we run the CHGRMTJRN command. The prompt display is shown in Figure 35.

We deactivate this remote journal using the immediate option. This command must be run on the source system.

```
                    Change Remote Journal (CHGRMTJRN)

Type choices, press Enter.

Relational database  . . . . . > TGTSYS
Source journal . . . . . . . . > XMPJRN        Name
  Library  . . . . . .           XMPLIB        Name, *LIBL, *CURLIB
Target journal . . . . . .     XMPJRN          Name, *SRCJRN
  Library  . . . . . .           XMPLIBBKP     Name
Journal state  . . . . . . . . > *INACTIVE     *SAME, *ACTIVE, *INACTIVE
How to make inactive . . . . . *IMMED          *CNTRLD, *IMMED

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 35. CHGRMTJRN Prompt Display—Deactivate Remote Journal

3.2.2 Using the CHGJRN Command for Deactivation

The CHGJRN command can be used to deactivate a remote journal on the system on which this journal resides. This is the same as what can be done using format CJST0200 on the QjoChangeJournalState API. When the CHGJRN command is used to deactivate a remote journal, every parameter on the command, except the following two, must be given a value of *SAME. The two parameters that can have a value other than *SAME are:

- **Journal (JRN)**

  This parameter specifies the name of the remote journal to deactivate.

- **Journal State (JRNSTATE)**

  Specify a key value of *INACTIVE to deactivate the remote journal. Deactivating a remote journal by using the CHGJRN command causes an immediate deactivation to occur. Journal entries that were queued up on the source system awaiting transmission are not sent.

While the CHGJRN command can be used to deactivate a remote journal, it cannot be used to activate one. Only the CHGRMTJRN command can be used for that purpose. To deactivate the remote journal that was activated in Figure 27 on page 35, run the CHGJRN command (shown in Figure 36 on page 46) on the target system where XMPJRN in XMPLIBBKP resides.

```
                              Change Journal (CHGJRN)

Type choices, press Enter.

Journal  . . . . . . . . . . . . JRN            XMPJRN
  Library  . . . . . . . . . . .                  XMPLIBBKP
Journal receiver:               JRNRCV
  Journal receiver . . . . . . .                *SAME
    Library  . . . . . . . . . .
  Journal receiver . . . . . . .
    Library  . . . . . . . . . .
Sequence option  . . . . . . . . SEQOPT         *CONT
Journal message queue  . . . . . MSGQ           *SAME
  Library  . . . . . . . . . . .
Manage receivers . . . . . . . . MNGRCV         *SAME
Delete receivers . . . . . . . . DLTRCV         *SAME
Receiver size options  . . . . . RCVSIZOPT      *SAME

Journal state  . . . . . . . . . JRNSTATE       *INACTIVE

                                                                        More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 36. CHGJRN Command Prompt Display Example

3.2.3 Which Command to Use

Under normal circumstances, use the CHGRMTJRN to deactivate a remote journal. This allows for a more orderly ending of the remote journal. However, since the CHGRMTJRN command must run on the source system, there are two conditions where the CHGJRN command must be used instead:

- The communications path that is referenced by the RDB entry is unavailable.
- The source system itself has failed.

Under either one of these conditions, it is not possible to use the CHGRMTJRN command. Instead the deactivation of the remote journal must be done on the target system using the CHGJRN command.

3.2.4 Deactivation Following System Failure

If the source system fails, any remote journal that was *ACTIVE at the time of the failure must be deactivated. The CHGJRN command must be run for each remote journal on each target system. When the source system fails, the remote journal is left in an active state. When the source system is available again, the source journal loses all record of any remote journals that were active at the time of failure. Unless action is taken, the source side believes all remote journals are inactive while each target side believes the remote journals are active. If you attempt to activate one of these remote journals, the CHGRMTJRN command fails.

Moreover, you must use the CHGJRN command. If you wait until the source system is available again and attempt to use the CHGRMTJRN command to deactivate it, it fails. Then, the remote journal is not deactivated.

If a target system fails, you do not need to take any action to deactivate the remote journals. In this case, the system failure event causes the active remote journals to be deactivated automatically.

3.2.5 Inactive Pending State

Under normal circumstances, a remote journal is in one of two states: *ACTIVE or *INACTIVE.

When activated, a remote journal's state changes from *INACTIVE to *ACTIVE. When deactivated, a remote journal's state directly moves from *ACTIVE to *INACTIVE under normal circumstances. However, there can be a situation where the state moves from *ACTIVE to a state referred to as *INACTPEND. This is an indicator that the remote journal is in the process of going to the *INACTIVE state. Under some error conditions, the process cannot be completed and the remote journal is left in the *INACTPEND state. While the remote journal is in an *INACTPEND state, it cannot be activated again. Under these conditions, you need to take action to bring the remote journal back into an *INACTIVE state.

This special state occurs only if the delivery mode is specified as synchronous when the remote journal is activated. The state means that the remote journal received replicated journal entries from the source system and could not confirm that reception back to the source system. This may be due to the fact that the source system, or communications to the source system, failed before the confirmation could be sent. It is a characteristic of synchronous delivery mode that journal entries are written first to the remote journal and then to the source journal only after confirmation is received by the source system. The changes are also not made to the database until the entries are written to the source journal. Thus, if the remote journal is in an *INACTPEND state it can contain journal entries that neither the source journal contains nor the source database reflects. These *unconfirmed* journal entries are an accurate reflection of what the application did on the source system, only the source system may have no record of their occurrence. You can view these unconfirmed entries and a high availability solution can apply these changes to the backup database if it so desired. If it does, the backup database may possibly be ahead of the production database. If there are unconfirmed journal entries in the remote journal, there is a message in the message queue associated with the remote journal. The message identifies how many unconfirmed entries are present.

To reset the state of a remote journal from *INACTPEND to *INACTIVE, the CHGJRN command must be run on the target system. The remote journal can be activated again. Note that the remote journal will not report back to the source system that it has unconfirmed entries. From a source perspective, these journal entries, and the transaction they represent, may be lost if the source system failed.

Attention

Before running the CHGJRN command to deactivate a remote journal from the *INACTPEND state, check the message queue to determine if there are unconfirmed entries present. Look for message CPF70D4. If it is present, the message tells you if there are any unconfirmed journal entries in the remote journal. Take any action on the unconfirmed entries first and then run the command. If you run the command first, the unconfirmed entries are lost.

3.3 Determining the Status of a Remote Journal

The Work with Journal Attributes (WRKJRNA) command can be used to view the status of either a local journal or a remote journal. The *QjoRetrieveJournalInformation* API provides the same type of information that WRKJRNA provides.

The starting point for viewing status should be the source journal. By running the WRKJRNA command against the source journal, you can see not only the status of that journal, but also the status of any added remote journals. However, it should be noted that WRKJRNA command only shows information for remote journals one level down or one level up in the local and remote journal hierarchy. If you have a multi-layer cascade configuration, you have to look on more than one system to see the whole network.

As an example, we ran the command shown in Figure 37 on the local journal on the source system.

```
WRKJRNA JRN(XMPLIB/XMPJRN)
```

Figure 37. WRKJRNA Example

When you run this WRKJRNA command, you see a display similar to the one shown in Figure 38. In this example, the source journal is a local journal. If the source journal is a remote journal, the display appears somewhat differently.

```
                        Work with Journal Attributes

Journal  . . . . . :   XMPJRN          Library  . . . . . :   XMPLIB

Auxiliary storage                      Journal type . . . . :   *LOCAL
  pool . . . . . :   1                 Journal state  . . . :   *ACTIVE
Message queue  . . . :   QSYSOPR       Receiver size options:   *NONE
  Library  . . . . . :     *LIBL
Manage receivers . . :   *SYSTEM
Delete receivers . . :   *NO
Text . . . . . :   *BLANK

Type options, press Enter.
  8=Display attributes

        Attached
Option  Receiver    Library
        RCVR0002    RCVLIB



                                                                      Bottom
 F3=Exit   F5=Refresh   F12=Cancel    F13=Display journaled files
F14=Display journaled access paths  F24=More keys
```

Figure 38. WRKJRNA Initial Display for a Local Journal

On the WRKJRNA display, the Journal type field identifies what the journal type is. The value here is either *LOCAL or *REMOTE. In this example, we are working with a local journal so the value is set to *LOCAL. The state of the journal

is given in the Journal state field. For a local journal, this value is *ACTIVE if the journal is active or *INACTIVE if it is not.

From the initial WRKJRNA display, you can obtain information about any added remote journals by pressing F16. If we do this for the example with which we are working, we see the display shown in Figure 39.

```
                    Work with Remote Journal Information

Journal  . . . . . :   XMPJRN          Library  . . . . . :   XMPLIB

Journal type . . . . :   *LOCAL        Journal state  . . . :   *ACTIVE
Remote journal type  :                 Delivery mode  . . . :
Local journal  . . . :                 Source journal . . . :
  Library  . . . . . :                   Library  . . . . . :
  System . . . . . . :                   System . . . . . . :
Redirected receiver library . . . . . . . . . . . . . . . . . :   RCVLIBBPK
Number of remote journals . . . . . . . . . . . . . . . . . . :   1

Type options, press Enter.
  5=Display remote journal details   8=Display relational database detail
  13=Activate   14=Inactivate
                          ---------------------Remote----------------------
     Relational                                      Journal      Delivery
Opt  Database             Journal      Library      State        Mode
__   TGTSYS               XMPJRN       XMPLIBBKP    *ACTIVE      *SYNC
                                                                       Bottom
===>
F3=Exit       F4=Prompt   F5=Refresh   F6=Work with remote journal list
F9=Retrieve   F12=Cancel
```

Figure 39. WRKJRNA Displaying Remote Journal Information for the Local Journal

The display that results from pressing F16 shows the added remote journals information at the bottom, while keeping the source journal information at the top. Each remote journal that is added to the source journal is displayed. For each remote journal displayed, you see:

- The RDB entry in the source system RDB directory that is used to connect to the target system on which the remote journal exists.
- The name of the remote journal.
- The name of the library that the remote journal is in on the target system.
- The state of the remote journal. This state can be one of the following:
  - *INACTIVE—The remote journal is inactive.
  - *ACTIVE—The remote journal is active and journal entries are replicated to it.
  - *FAILED—The remote journal cannot receive journal entries from the source journal due to an error condition.
  - *CTLINACT—The remote journal is in the process of a controlled inactivate request.
- If the remote journal is active, the delivery mode that is used to replicate journal entries. The value here can be:

– *ASYNC—The delivery mode is asynchronous.

– *SYNC—The delivery mode is synchronous.

– *ASYCNPEND—The system support is currently in the catch-up phase for the remote journal. Once it is completed, the delivery mode is asynchronous.

– *SYNCPEND—The system support is currently in the catch-up phase for this remote journal. When it is complete, the delivery mode is synchronous.

Note

From the Work with Remote Journals display, you can activate or deactivate a remote journal by using option 13 or 14. Using one of these options and pressing F4 causes a prompt display for the CHGMRTJRN command to appear. Otherwise, you can just press enter to use the defaults.

If detailed information is wanted about a particular remote journal, using option 5 provides that information. In our example, if we use option 5, we see the display shown in Figure 40. On this display, you see additional information such as the type of the remote journal and the receiver library redirection.

```
                    Display Remote Journal Details

Remote Journal . . . :   XMPJRN         Library  . . . . . :   XMPLIBBKP

Relational database  . . . . . . . . . . . . :   TGTSYS
Remote journal type  . . . . . . . . . . . . :   *TYPE1
Remote journal state . . . . . . . . . . . . :   *ACTIVE
Remote delivery mode . . . . . . . . . . . . :   *SYNC
Remote journal receiver library  . . . . . . :   RCVLIBBPK









Press Enter to continue.


F3=Exit   F11=Display relational database detail   F12=Cancel
```

Figure 40. Remote Journal Detail Display

If you use option 8, you can see more information about the Relational Database Directory Entry that is currently associated with the remote journal.

If the WRKJRNA command is used for a remote journal rather than a local journal, the first display that appears as shown in Figure 41 on page 51. Notice that you now see information that only pertains to a remote journal such as the Remote journal type field. Also, for a remote journal, the state can be other than *ACTIVE or *INACTIVE as mentioned previously.

Notice also that the origin of each journal receiver is given. The first field, Local Journal System, identifies the system on which the origin local journal exists. The second field, Source Journal System, identifies the system that the journal entries are being replicated from to this remote journal. This is important in a cascade configuration. In our example, we do not have a cascade configuration so both fields identify the same system. If the remote journal you are working with is also the source for another remote journal, pressing F16 allows you to view information about the remote journals that are added to the one with which you are working.

```
                        Work with Journal Attributes

Journal  . . . . . :   XMPJRN          Library  . . . . . :   XMPLIBBKP

Auxiliary storage                      Journal type . . . . :   *REMOTE
  pool . . . . . :   1                 Journal state  . . . :   *ACTIVE
Message queue  . . . :   QSYSOPR       Remote journal type  :   *TYPE1
  Library  . . . . . :     QSYS        Receiver size options:
Manage receivers . . :
Delete receivers . . :   *NO
Text . . . . . :   *BLANK

Type options, press Enter.
  8=Display attributes
                                 Local     Source
       Attached                  Journal   Journal
Option Receiver    Library       System    System
  _    RCVR0002    RCVLIBBPK     SRCSYS    SRCSYS


                                                                    Bottom
F3=Exit   F5=Refresh   F12=Cancel    F13=Display journaled files
F14=Display journaled access paths   F24=More keys
```

Figure 41. WRKJRNA with Remote Journal

If you are in a high availability environment, one important piece of status information is not readily available. To ensure that the backup system remains current, you need to know how far behind the remote journal lags the source journal. If it lags significantly behind, your backup system is not current with your production system. Unfortunately, there is no direct way to determine this. The only way is to view the status of the currently attached receiver on the source journal and compare its status to the currently attached receiver on the remote journal. By comparing the sequence number of the last entry in each receiver, you get an approximate idea about how far behind the remote journal lags.

To view the status of the attached receivers, perform the following steps:

1. Run the `WRKJRNA` command on the source system for the source journal.
2. Run the `WRKJRNA` command on the target system for the remote journal.
3. Select option 8 on each of the Work with Journal Attributes displays to view the status of the currently attached receivers.
4. Compare the values in the Last Sequence Number field on each display. The last sequence number in the receiver attached to the remote journal is equal to or behind the last sequence number in the receiver attached to the source

journal. The difference between the sequence number is an approximate measure of how far the remote journal lags behind the source journal.

Note

The *QjoRtrvJrnReceiverInformation* API provides program interfaces to the DSPJRNRCVA information for a journal receiver. This is the information shown when option 8 is selected from WRKJRNA command.

3.4 Managing Journal Receivers

Managing journal receivers involves two tasks:

- Attaching a new receiver to a local journal when required.
- Deleting old receivers when they are no longer needed.

When using the remote journal function, the first task applies to the local journal only. You cannot directly cause a new receiver to be attached to any remote journal. Any attempt to do so, for example, through the CHGJRN command, results in an error, and the currently attached receiver is not changed. The system support automatically handles this task for remote journals. When a new journal receiver is attached to the local journal, the system support automatically attaches that receiver to each remote journal that is currently connected to that local journal. This occurs when the system prepares to replicate journal entries in the new receiver to a remote journal. For a remote journal that is already active, this can happen immediately. For a remote journal that is inactive, this occurs the next time it is activated.

From a remote journal perspective, it makes no difference what strategy you choose to attach new receivers to the local journal. You may do it directly by using the CHGJRN command, or you can allow the system to automatically change the receiver when needed. The end result to any connected remote journal is the same. The method you use to attach a new receiver to the local journal is initially set when you create the local journal through the CRTJRN command (MNGRCV parameter). After the local journal is created, you can change this attribute by using the CHGJRN command (MNGRCV parameter).

The second task applies to both the local and remote journals. For deletion purposes, you need to manage the local journal's receiver chain and each remote journal's receiver chain. Once a receiver is created on the receiver chain of a remote journal, it continues to exist independent of the local receiver. That is, deleting a receiver from the local journal chain has no impact on the receiver chain of any remote journal. You must manage each chain individually.

The first decision to make is to determine what delete strategy you use. You can allow the system to delete receivers automatically when they are no longer needed, or you can take on the responsibility and delete receivers directly. You can use a mixed strategy where the system deletes receivers automatically from the local journal, but you directly control the deletion of receivers for one or more remote journals. Once you decide, you must set this delete receiver attribute appropriately for the local journal and for each remote journal that is linked to that local journal.

The delete receiver attribute for the local journal is set initially when you create the journal through the CRTJRN command. Using the DLTRCV parameter, you specify *YES if you want the system to automatically delete receivers when they are no longer required. You specify *NO if you do not want the system to automatically delete receivers. You may change this specification after the journal is created by using the CHGJRN command (using the DLTRCV parameter again). For a local journal, you can only specify DLTRCV(*YES) if the MNGRCV parameter is set to *SYSTEM.

The delete receiver attribute for a remote journal is initially set when the ADDRMTJRN command (from the DLTRCV parameter) is run to add a new remote journal to a source journal. The delete receiver attribute for the remote journal does not have to be the same as the source journal. For example, the source journal may specify *NO for the delete receiver attribute while the remote journal can specify *YES. Moreover, if there is more than one remote journal linked to the same source journal, the delete receiver attribute does not have to be the same across them. Once the remote journal is added, the delete receiver specification can be changed for the remote journal by running the CHGJRN command on the target system.

Note

If you remove the link between a remote journal and a source journal using the RMVRMTJRN command, no objects are actually removed. Only the link is removed. If you run another ADDRMTJRN command for that remote journal, no new objects are created. Rather, the link is re-established between the source journal and the pre-existing remote journal. In this case, the delete receiver attribute is left unchanged in the remote journal. The specification you gave on the second ADDRMTJRN command is ignored.

For example, we want to create a local journal where we attach new receivers and delete old receivers. To the local journal, we add a remote journal where we want the system to automatically delete receivers when they are no longer needed. To create this configuration, we run the CL commands shown in Figure 42.

```
CRTJRN JRN(XMPLIB/XMPJRN)   +
                JRNRCV(RCVLIB/RCVR0001)  +
                MNGRCV(*USER)  +
                DLTRCV(*NO)
ADDRMTJRN RDB(TGTSYS)   +
                SRCJRN(XMPLIB/XMPJRN)  +
                TGTJRN(XMPLIBBKP/XMPJRN)  +
                RMTRCVLIB(RCVLIBBKP)  +
                RMTJRNTYPE(*TYPE1)  +
                DLTRCV(*YES)
```

Figure 42. Delete Receiver Management—CL Commands

Running the previous two commands and activating the remote journal using CHGRMTJRN command results in the remote journal configuration shown in Figure 43 on page 54.

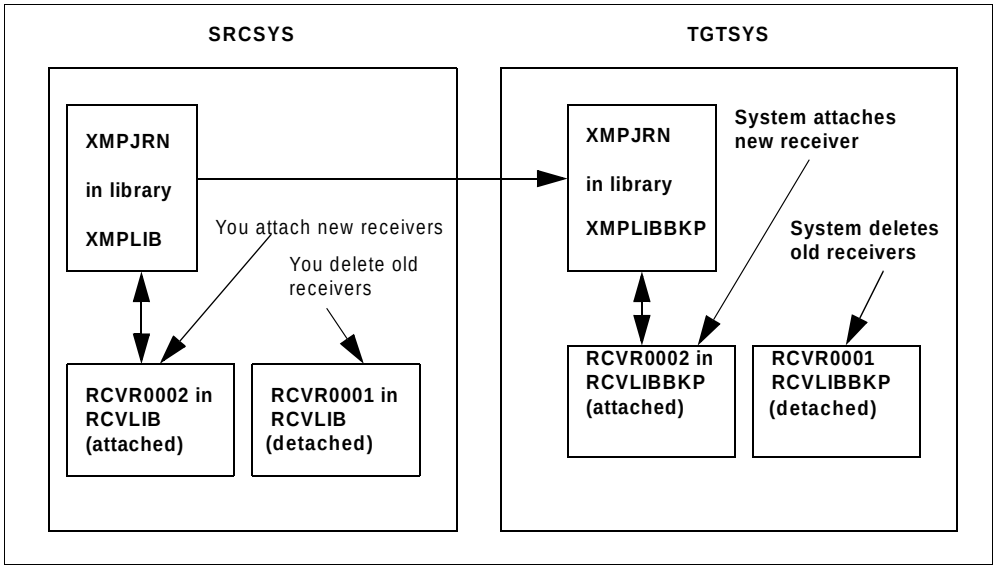


Figure 43. Example of Receiver Management

If we later decide that we do not want the system to automatically delete the remote journal receivers, we run the CHGJRN command (as shown in Figure 44) on the system TGTSYS.

```
CHGJRN  JRN(XMPLIBBKP/XMPJRN)  DLTRCV(*NO)
```

Figure 44. Delete Receiver Management—Changing the Current Choice

3.4.1 Deleting Source Journal Receivers

If you elect to perform the deletion of local receivers manually rather than allowing the system to do it automatically, you need to use the DLTJRNRCV command.

The DLTJRNRCV command provides a level of protection to prevent you from inadvertently deleting a receiver that is still needed. Please note that system managed deletes provide the same level of protection. There are two important checks that are made for a source journal receiver:

- A check is made to determine if the receiver is fully saved. If it is not, the system sends an inquiry message so the user can confirm that the delete must be carried out.
- A check is made to ensure that the receiver is fully replicated to all remote journals that are currently added to the source journal.

On the DLTJRNRCV command, you can specify to ignore the two checks. If you specify to ignore the checks, you must be certain that you indeed want to delete the receiver. The system does not give you any indication that the receiver is not saved or replicated.

With a remote journal, you may have a save or restore strategy that saves receivers off of the remote system rather than the source system. Therefore, when deleting source journal receivers, you can safely ignore the system save

check. However, if you save receivers from the source system, do not consider ignoring the system check on the save information.

Do not ignore the second check, whether the receiver is replicated, under normal operating procedures. In the simplest remote journal configuration, it is not readily apparent whether a given receiver on the source system is replicated to the target system. You must go to the target system and examine the remote journal to see if the receiver is fully replicated. In more complex environments where there are several remote journals added to a source, you must examine each one before you are sure. Different remote journals may be activated at different times. As a result, they are in different states relative to what each has in its receiver chain. Moreover, deleting a source journal receiver before it is replicated can have severe results. For example:

- If your save strategy is to save receivers off of the target system, deleting the source journal receiver before it is replicated means the data is not available to be saved.
- If you are using the remote journal for high availability purposes, deleting the source journal receiver before it is replicated means that the backup system cannot be kept current with the production system. The transactions represented by the journal entries in the deleted receiver cannot be applied to the backup database. After such an event, bringing the backup database to the same level as the production database can be time consuming, possibly even to the extent of requiring a complete restore of the database.

The rules governing whether a receiver needs to be replicated before deletion are complex. In general, if a receiver is not attached to the source journal at the time a remote journal is added, the receiver is viewed as not requiring replication before deletion. It may be replicated. However, if you attempt to delete the receiver first, the system will not prevent it. All receivers that are attached to the source journal while the remote journal is added are protected, unless you specify DLTOPT(*IGNTGTRCV) on the DLTJRNRCV command. If more than one remote journal is added to a source journal, but at different points in time, some journal receivers are protected until they are replicated to one remote journal. Others are protected until they are replicated to all added remote journals. Similarly, if a remote journal is added and removed, receivers detached from the source journal after the remote journal was removed are no longer protected.

To illustrate the deletion protection rules, consider Figure 45 on page 56. In this example, the source journal only has one receiver on its receiver chain, RCVR0001. If a remote journal is added to the source journal, RCVR0001 and all new receivers that are attached from that point on are protected from deletion until replicated to the remote journal. Assume that the receiver chain on the source journal now consists of two receivers RCVR0002, which is the attached receiver, and RCVR0001.

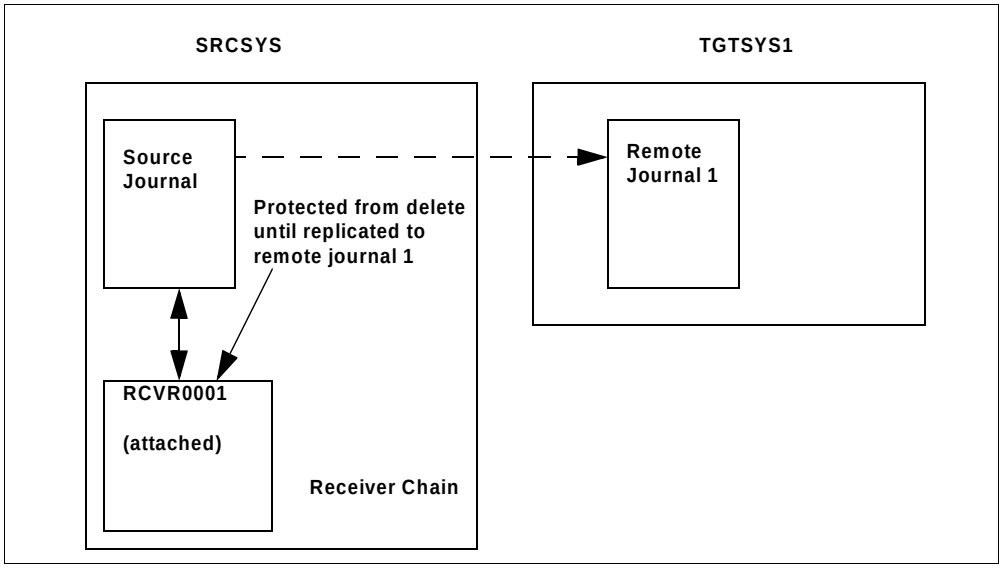


Figure 45. Delete Protection Rules (Part 1)

At this point, we add a second remote journal as shown in Figure 46. Now RCVR0002 and RCVR0001 are both protected from deletion until they are replicated to the first remote journal. However, only receiver RCVR0002 is protected from deletion until it is replicated to the second remote journal. RCVR0001 is not protected because it was detached from the source journal before the second remote journal was added. Thus, if RCVR0001 is replicated to the first remote journal but not the second, it can be deleted successfully.

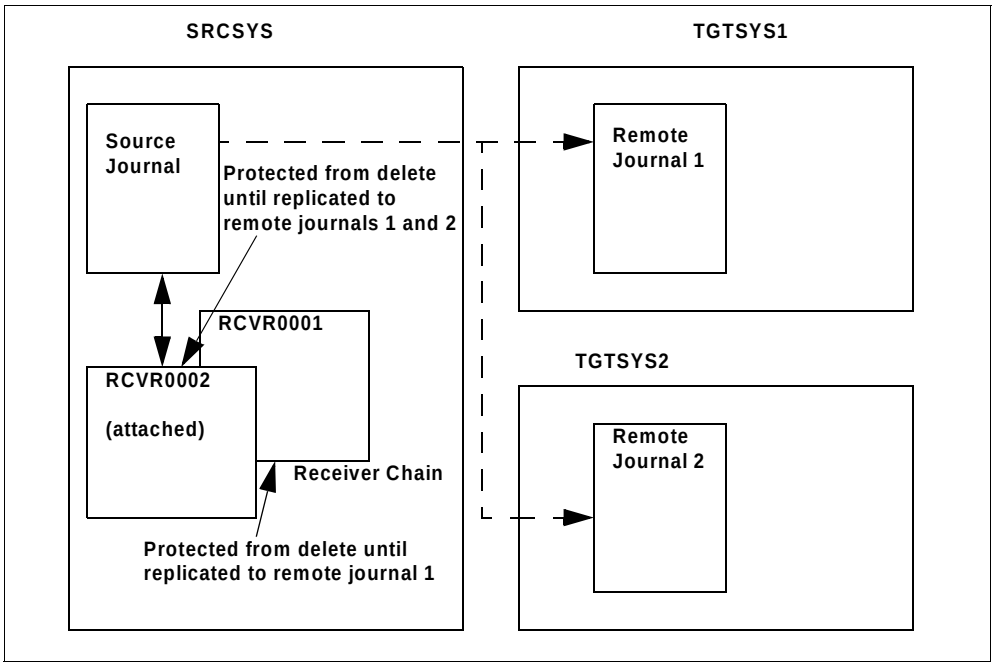


Figure 46. Delete Protection Rules (Part 2)

3.4.2 Deleting Remote Journal Receivers

Manual deletion of receivers on the remote journal's chain is also accomplished by using the DLTJRNRCV command. With a remote journal, the same level of protection is provided to prevent you from deleting a receiver inadvertently while it may still be needed:

- A check is made to determine if the remote journal receiver is saved yet. If not, an inquiry message is sent so you can confirm that deletion is to occur.
- If the remote journal is also a source journal in a cascade configuration, a check is made to determine whether the receiver that is deleted is replicated to next remote journal in the configuration. If not, an error message is sent, and the delete fails. If the remote journal is not also a source journal, this check is not applicable.

As in the case of source journal receivers, you can select to ignore the above two checks using a parameter on the command. The same precautions as previously mentioned for the source journal apply here. Do not prevent the inquiry message from being sent unless you are absolutely sure the receiver can be deleted without damage to your save and restore strategy. If you do not allow the inquiry message on escape exception to be sent, you will not have the opportunity to correct a mistake. Similarly, do not specify to ignore the target receiver on the DLTJRNRCV command if deleting it will affect your high availability solution.

The Delete Journal Receiver exit point, QIBM_QJO_DLT_JRNRCV, may be of assistance as well. You may want to register an exit program with this exit point, which will be called every time a journal receiver is deleted from the system. This program can be used to verify that the journal receiver is eligible for deletion. Refer to *Backup and Recovery*, SC41-5304, for more details about this exit point.

If you have a cascade configuration, the rule governing whether a receiver can be deleted before being replicated gets even more complex. The rule is the same. However, it must be applied to each leg in the cascade configuration and is best explained through an example.

We begin again with Figure 45 on page 56. We activate the remote journal and allow journal entries to be replicated to the remote journal, resulting in the configuration shown in Figure 47 on page 58. We create a cascade configuration by adding a second remote journal on TGTSYS2 to the first remote journal on TGTSYS1. This results in the configuration shown in Figure 48 on page 59.

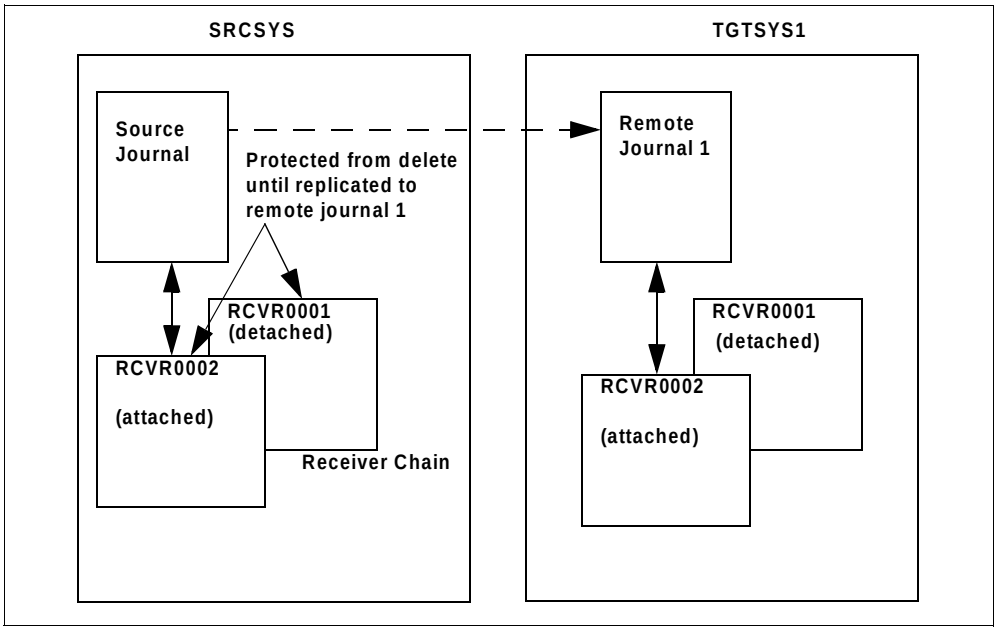


Figure 47. Delete Rules Cascade Configuration (Part 1)

Notice the following points:

- All source journal receivers on the system SRCSYS are protected from deletion until replicated to the first remote journal on TGTSYS1.
- Only RCVR0002 on TGTSYS1 is protected from deletion until replicated to remote journal 2 on TGTSYS2. RCVR0001 on TGTSYS1 is not protected because it was detached from remote journal 1 before remote journal 2 was added.

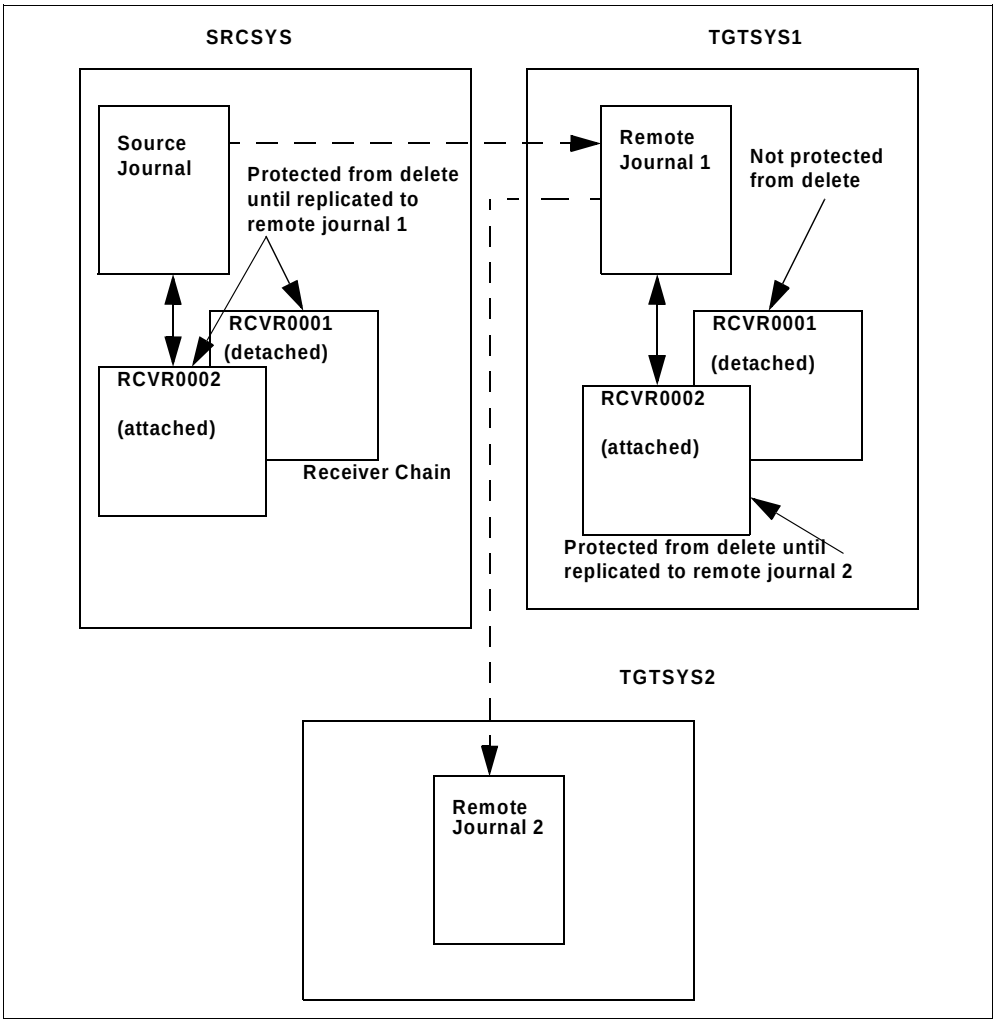


Figure 48. Delete Rules Cascade Configuration (Part 2)

Note: If your environment is simple, remember this simple rule for the receiver deletion: You cannot delete a receiver until it is fully replicated to all immediate downstream remote journals that are added to this source journal.

3.5 Managing Multiple Remote Journals

The remote journal examples shown up to this point are simple and easy to manage manually. However, the system support allows you to create complex remote journal configurations. You may create broadcast or cascade configurations that involve several systems and remote journals. You may create hybrid configurations where part is broadcast and the remainder is cascade. As the number of remote journals increases, the job of managing the remote journals grows proportionally. Each *source/remote journal pair* must be *managed individually*. Action taken against one pair has no impact on any other pair. At some point, manually managing the remote journals may prove too difficult and too prone to error. A program-based procedure may become necessary to automate at least some of the management tasks.

The objective of this chapter is to highlight the tasks that need to be performed with multiple remote journals. This is done through the example shown in Figure 49.

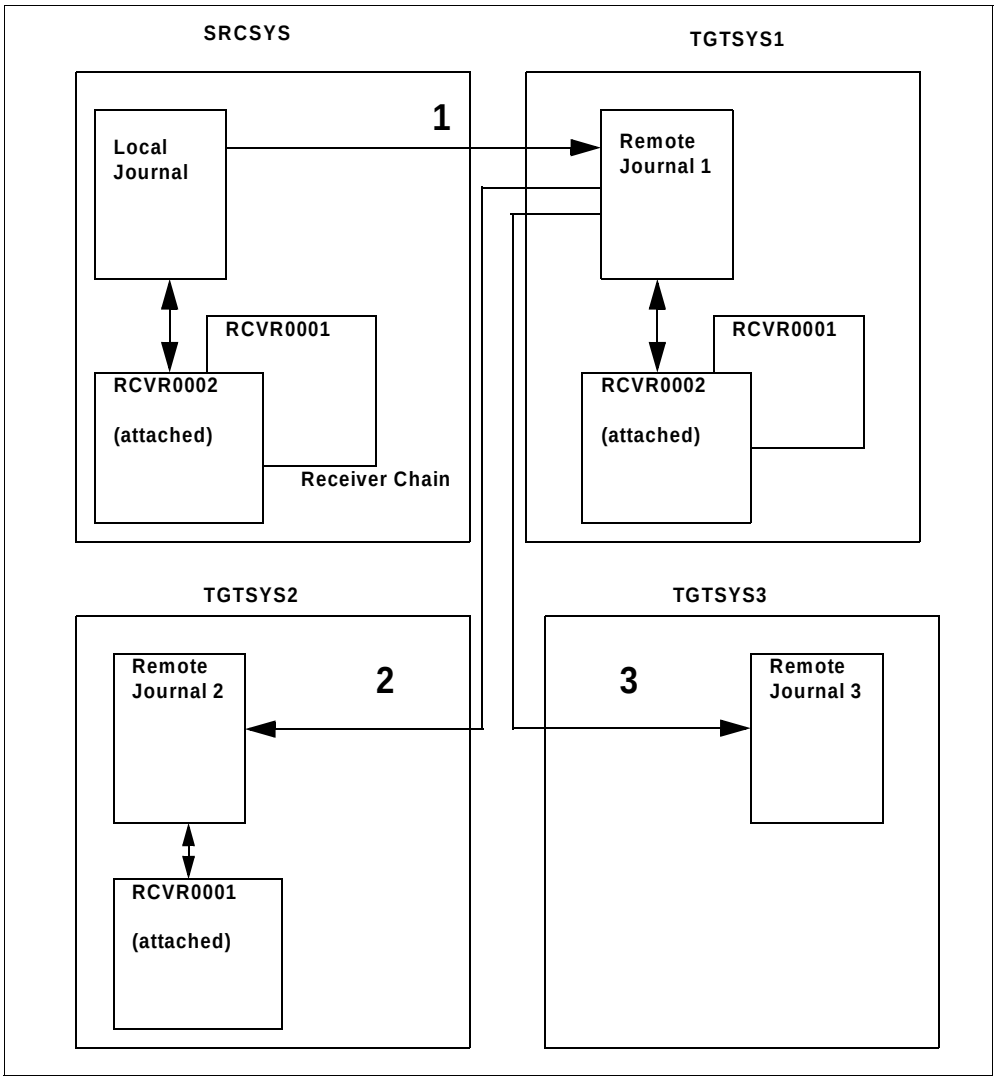


Figure 49. Example of Multiple Remote Journals

Observe the following points from this example:

- There are three source and remote journal pairs that must be managed across four different systems. The pairs are labeled 1, 2, and 3.
- Each pair must be activated or deactivated individually. In this example, pairs 1 and 2 are activated (as represented by the solid line), but not pair 3. As a result, journal entries are replicated to two remote journals but not the third. The decision to not activate the third pair may be deliberate or may result from an oversight. In either case, remote journal 3 is not maintained at a level current with the other remote journals. A similar result is seen if the pairs are not deactivated in the correct order.
- Activating pair 1 is critical to replicating to pairs 2 and 3. New journal entries originate from the source journal on SRCSYS. If pair 1 is not active, the new

journal entries cannot be replicated to remote journal 1. As a consequence, they cannot be replicated to any other remote journal downstream from it. Conversely, deactivating pair 1 allows remote journals 2 and 3 to catch up with 1, but prevents new journal entries that are deposited in the source journal from being replicated.

- Activating all pairs using the same starting point can be critical. If you do not use the same starting point consistently, the journal entries replicated to each system will be inconsistent.
- The delivery mode for pair 1 can be either asynchronous or synchronous. The delivery mode for pairs 2 and 3 can only be asynchronous. If you want to use synchronous delivery mode to enhance your high availability solution, ensure that your high availability system with its remote journal is immediately downstream from the source system.
- To determine the status of each remote journal, you need to look at each pair. Looking at pair 1, for example, does not give you a complete view of pairs 2 and 3.
- If you manually delete the receivers, you need to manage four receiver chains, one on each system. This is one case where depending on the system to indicate whether a receiver is eligible for deletion because it is not replicated to all remote journals is useful.

Figure 49 represents only one possible remote journal configuration. Many more complex configurations can be created. The points mentioned above apply to other configurations. Keep these points in mind as you add new remote journals and systems.

3.6 Monitoring for Errors

Once a remote journal is activated, errors can result or be detected on either the source side or the remote side. Both the source journal and the remote journal have an associated message queue. Errors that originate on the source side result in messages being sent to the message queue that is associated with the source journal. Errors that originate on the remote side result in messages being sent to the message queue associated with the remote journal. In addition, errors that originate on the remote side are, in most cases, reported back to the source side. This, in turn, causes a message to be sent to the message queue associated with the source journal. Please refer to the chapter "Remote Journal Function" in *OS/400 Backup and Recovery*, SC41-5304, for a list of the remote journal function messages.

For errors originating on the remote side, the message that is sent to the message queue associated with the remote journal is more detailed than the message that is sent to the message queue associated with the source journal. A good strategy for monitoring for errors is to monitor the message queue on the source side. If a message is seen that relates to the remote side, you can view the messages in the message queue on the remote side to obtain additional information.

3.7 Recovering from Communication Failure

If the communications path fails between the source and remote journal, replication of the journal entries stops. An error message is sent to the message queue associated with the source journal. First, ensure that the remote journal is deactivated. Run the WRKJRNA command on the remote system to determine the status of the remote journal there. Normally on a communications failure, the remote journal is inactivated by the system automatically. However, if you were running in synchronous delivery mode, the remote journal may be left in an *INACTPEND state. You need to run the CHGJRN command on the target system to inactivate the remote journal, but only after you have deal with the unconfirmed changes. Refer to 5.3.2, “Replaying the Unconfirmed Journal Entries” on page 86, for more details on unconfirmed changes.

If the communication path that failed is the only path between the source and remote system, you need to wait until the problem is corrected before journal entry replication can continue. If it is not the only path, you may continue replication by using the alternative path. For example, you may be using OptiConnect/400 for replication because of performance reasons, and both the source and remote system are also connected using an Ethernet LAN. If OptiConnect/400 fails, you can continue replication over the Ethernet.

Continuing on an alternative path is easy assuming that you prepared for the event ahead of time. The system allows you to run the ADDRMTJRN command multiple times for the same source/remote journal pair. The only difference between the runs of the command is that the value for the RDB parameter is different. In this example, we run the ADDRMTJRN command twice. The first time, we specify the RDB entry that represents the OptiConnect/400 path. The second time, we use the RDB entry that represents the Ethernet LAN path.

Under normal operating conditions, we activate the remote journal using the CHGRMTJRN command. Specify the RDB entry that represents the OptiConnect/400 path. Upon failure of this connection, we first ensure that the remote journal is deactivated. Then, we run the CHGRMTJRN command again to activate the remote journal, but this time, we specify the RDB entry for the Ethernet connection. Now journal entry replication that previously flowed over the OptiConnect/400 connection starts to flow over the Ethernet connection instead.

Attention

When the alternative communications path is slower than the primary, the delivery mode may have to be changed when the remote journal is activated using the alternative path. For example, if the synchronous delivery mode is used with OptiConnect/400, when you switch to use the Ethernet alternative path you may want to also switch the delivery mode to asynchronous. Leaving the delivery mode as synchronous may cause unacceptable degradation to the application because the speed of Ethernet is significantly less than OptiConnect/400. Refer to Chapter 4, “Remote Journal Performance” on page 69, for more information on remote journal function performance.

To summarize, the steps to recovering from a communications failure include:

1. **Preparation**—Run the ADDRMTJRN command for each communications path that you may potentially use between the source and remote system for journal entry replication.

   In our example, we run the command twice. The first time, we run it for the OptiConnect/400 path. The second time, we run it for the Ethernet LAN path. The CL commands to use are shown in Figure 50. Note that the RDB names were originally created to indicate the system name and the protocol.

```
ADDRMTJRN RDB(OPCTGTSYS) +
                    SRCJRN(XMPLIB/XMPJRN) +
                    TGTJRN(XMPLIBBKP/XMPJRN) +
                    RMTRCVLIB(RCVLIBBKP) +
                    RMTJRNTYP(*TYPE1) +
                    DLTRCV(*NO) +
                    TEXT('Remote journal over OPC/400')
ADDRMTJRN RDB(ETHTGTSYS) +
                    SRCJRN(XMPLIB/XMPJRN) +
                    TGTJRN(XMPLIBBKP/XMPJRN) +
                    RMTRCVLIB(RCVLIBBKP) +
                    RMTJRNTYP(*TYPE1) +
                    DLTRCV(*NO)
```

Figure 50. Preparing for Recovery

2. **Normal Operations**—Activate the remote journal using the communications path for normal operations.

   In our example, we run the CHGRMTJRN command specifying the RDB entry for the OptiConnect/400 path. We select the delivery mode as synchronous since we want to enhance our high availability solution. The speed of OptiConnect/400 permits us to do this without adverse performance affects to our applications. The CL command used is shown in Figure 51.

```
CHGRMTJRN RBD(OPCTGTSYS) +
                    SRCJRN(XMPLIB/XMPJRN) +
                    TGTJRN(XMPLIBBKP/XMPJRN) +
                     JRNSTATE(*ACTIVE) +
                     DELIVERY(*SYNC) +
                     STRJRNRCV(*ATTACHED)
```

Figure 51. Normal Operations—Activating Remote Journal

3. **Communications Failure**—Ensure the remote journal is deactivated.

   In our example, we need to check the status of the remote journal on the target system since we are running with the synchronous delivery mode. If the state of the remote journal is not inactive, we deactivate it by running the CHGJRN command on the target system as shown in Figure 52.

```
CHGJRN JRN(XMPLIBBKP/XMPJRN) +
                     JRNSTATE(*INACTIVE)
```

Figure 52. Deactivating Remote Journal Following Failure

4. **Recovery**—Activate the remote journal again, on the source system this time specifying the RDB entry for the alternative communications path.

   In our example, we run the CHGRMTJRN command again. This time we specify the RDB entry for the Ethernet LAN connection. Because the Ethernet connection is substantially slower than OptiConnect/400, we use the asynchronous delivery mode when we activate this time as shown in Figure 53.

```
CHGRMTJRN RBD(ETHTGTSYS) +
                    SRCJRN(XMPLIB/XMPJRN) +
                    TGTJRN(XMPLIBBKP/XMPJRN) +
                     JRNSTATE(*ACTIVE) +
                     DELIVERY(*ASYNC) +
                     STRJRNRCV(*ATTACHED)
```

Figure 53. Activating Remote Journal on Alternative Path

3.8 Save and Restore Considerations

If you are using a remote journal, you have a choice in defining your save and restore strategy for journal receivers. In addition, if you have a high availability solution that keeps your backup database updated and current with your production database, you have a choice in defining your save and restore strategy for the database itself.

For journal receivers, you may save them from the local journal on the original source system or from a remote journal on any of the remote systems. Saving receivers from the remote journal has several advantages:

- It allows you to off load work from the original source system, which is usually busy, to a backup system, which is usually not so busy.
- Receivers saved from a remote journal can be restored back to the remote journal they were saved from, as well as to the local journal on the original source system. As a result, one set of saved receivers can be used for recovery purposes for both the local and remote journals.
- If you have a network of remote journals and all are of *TYPE1, you may save the receivers from any one of the remote journals and restore those receivers back to any other remote journal as well as to the original local journal.

Before linking the first remote journal, decide what the save and restore strategy will be. If you base that strategy on remote journals, you must understand the difference between *TYPE1 and *TYPE2 remote journals. Using only *TYPE1 remote journals gives you the greatest flexibility for your save and restore strategy, but limits your naming conventions for your remote journals. Conversely, using *TYPE2 remote journals gives you the greatest flexibility in naming your remote journals but can limit your save and restore strategy. The concept of these different types of remote journals was introduced previously in 1.3.4, “Remote Journal Types” on page 8.

From a save and restore strategy, the difference between *TYPE1 and *TYPE2 journal receivers becomes a matter as to where you can restore the saved receivers. With a *TYPE1 remote journal, you can restore the saved receivers to:

- The remote journal from which the receivers were originally saved
- The local journal from which the journal entries were originally replicated
- Any other *TYPE1 remote journal that is receiving replicated journal entries from the same local journal

With a *TYPE2 remote journal, you are more limited. If you save receivers from a *TYPE2 remote journal, you can restore them to:

- The remote journal from which the receivers were originally saved
- The local journal from which the journal entries were originally replicated

You cannot restore the receivers back to any other remote journal.

The save and restore differences between a *TYPE1 and a *TYPE2 remote journal is illustrated in Figure 54 on page 66. In this example, remote journals 1 and 3 are *TYPE1 remote journals. Remote journal 2 is a *TYPE2 remote journal. If we save receivers from remote journal 1, we can restore them to remote journal 1, remote journal 3, or the original local journal. If we save receivers from remote journal 2, we can only restore them to remote journal 2 and the original local journal. We cannot restore them to remote journal 1 or remote journal 3.

If your use of remote journal is simple with only one remote journal linked to a local journal, the question of *TYPE1 or *TYPE2 is not important. In this case, either works satisfactorily.

If you use a high availability solution to receive journal entries from a remote journal on a backup system and apply the corresponding changes to a backup database, you can choose how you save your database. You may save it from either the production system or from the backup system. Saving from the backup has several advantages:

- You off load a significant workload from the production system to the backup.
- Saving the database is not disruptive to applications. Applications on the production system can continue to run, while a save of the backup database is performed. Scheduling time for a save is easier. You do not have to contend with the problems of saving while active if you trying to keep your applications running while doing a save.
- The remote journal can remain active while the save of the backup database is ongoing. The apply function of the high availability solution needs to be ended. Therefore, changes are still captured by the backup.

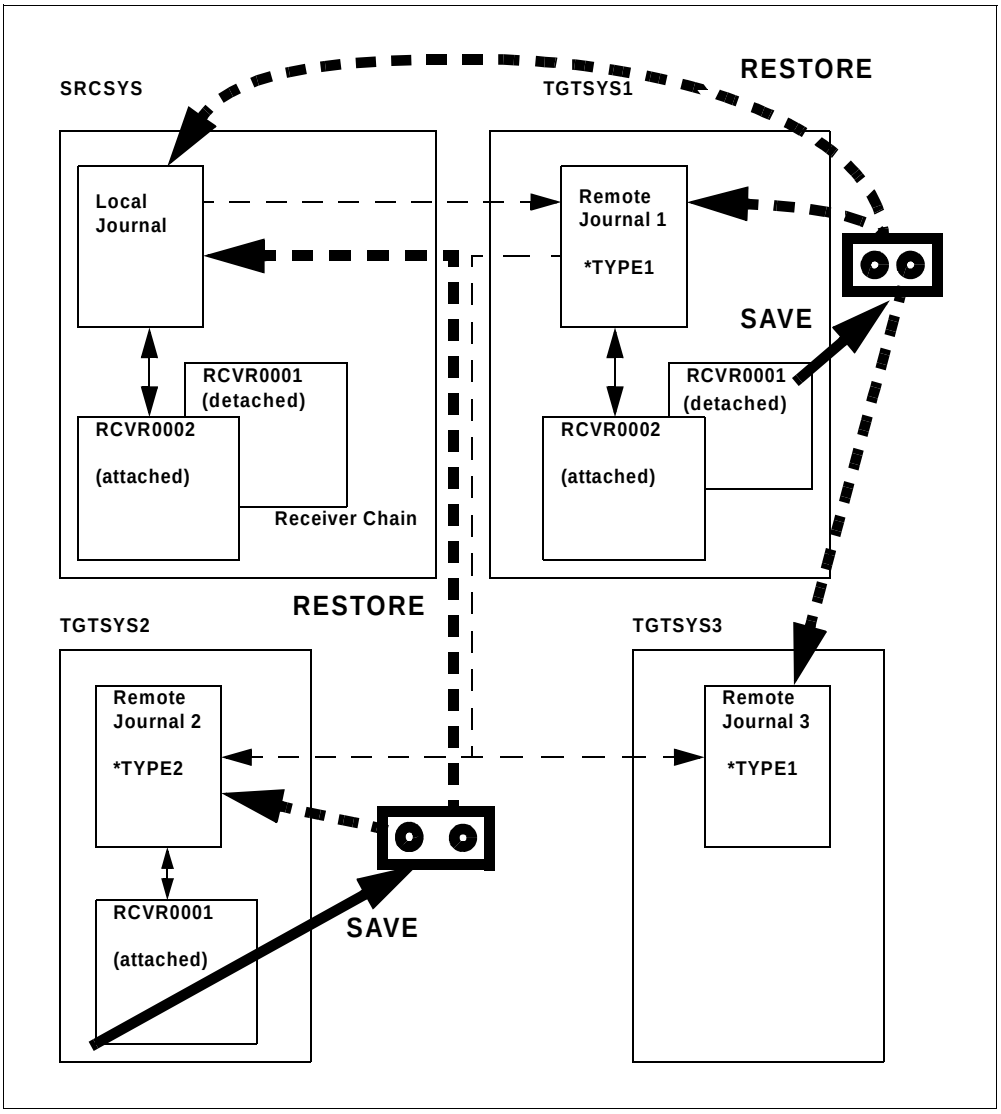


*Figure 54. Save and Restore *TYPE1 versus *TYPE2*

3.9 High Availability Considerations

In a high availability environment, journal entries need to be replicated in different directions depending upon whether you are running your applications on the production system or have done a switchover to the backup system. While you are running on the production system, journal entries are replicated from the local journal on the production system to a remote journal on the backup system. However, when the production system is unavailable, either because of a planned or unplanned shutdown, the application runs on the backup system. The local journal is on the backup system. Journal entries need to be replicated to a remote journal on the production system to capture changes that are made to the database while the production system is unavailable. As a result, high availability requires the use of two local and remote journal pairs.

A typical high availability environment is shown in Figure 55 on page 67. In this figure, the boxes with a solid outline represent functions that are active when you

are running on the production system and replicating journal entries to the backup system. The boxes with a broken outline represent functions that are active when you are running on the backup system while the production system is unavailable.

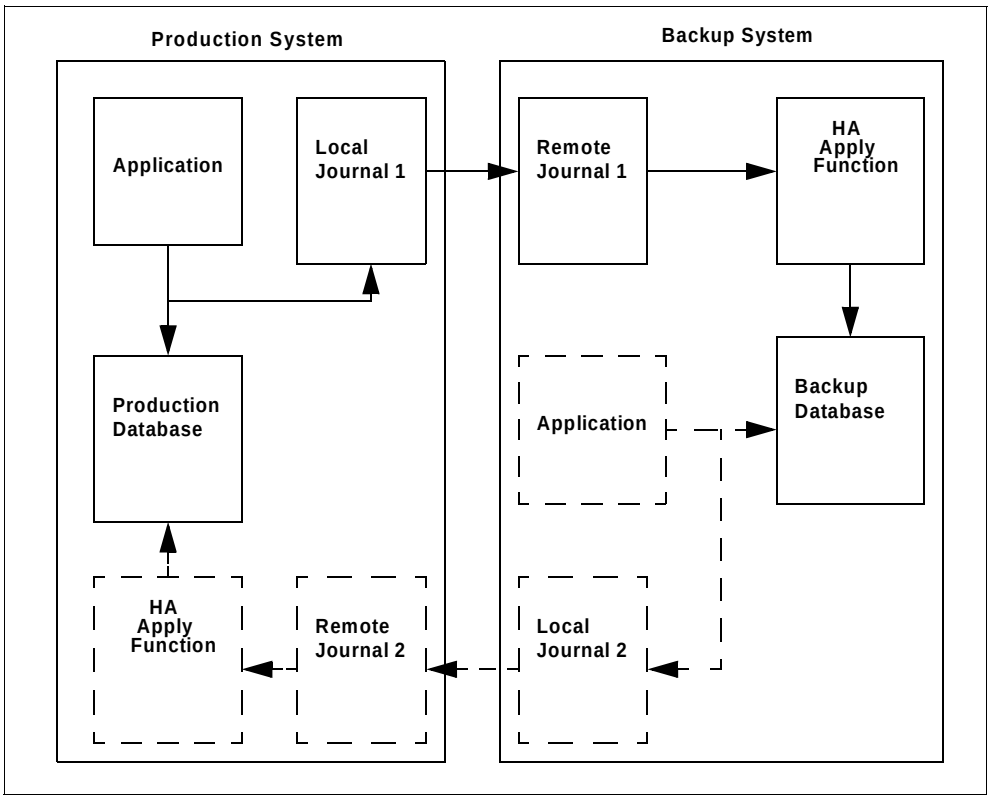


Figure 55. High Availability Remote Journal Example

When you run on the production system, the operations that occur are:

1. The application running on the production makes changes to the production database that causes a journal entry to be deposited in local journal 1.
2. The remote journal system support replicates journal entries that are deposited in local journal 1 to remote journal 1.
3. The Apply function of your high availability solution reads the replicated journal entries from remote journal 1 and makes the same changes to the backup database. This keeps the backup database current with the production database.

When you switch over to the backup system, the following operations occur:

1. The application runs on the backup system working with the backup database. Changes made to the backup database cause a journal entry to be deposited into local journal 2.
2. Since the production system is unavailable, remote journal 2 cannot be activated at this time. The journal entries are simply held pending in local journal 2.

3. When the production system is once more available, remote journal 2 is activated. The remote journal system support will now replicate all journal entries that were held pending over to remote journal 2.
4. The Apply function of your high availability solution is started on the production system. It reads the journal entries from remote journal 2 and makes the same changes to the production database, which brings it up to the same level as the backup database.

When managing remote journals in a high availability environment, as was previously described, consider the following precautions:

- Be sure that you do not delete local receivers until they are fully replicated. Do not delete remote receivers before the Apply process fully reads them.
- Before switching to the new system, allow the Apply function to completely process all receivers on the remote journal before you start your applications on that system. If you end the Apply function before it completes or start your applications prematurely, you may end up working with a down-level database.
- Ensure that you start physical file journaling on your backup system before you start your applications. Capture all changes to the database on that system so the same changes can be replicated to the production system.
- Before you start physical file journaling on the new system, attach a new receiver to the local journal. This ensures that any journal entries that were deposited after the switch occurred start at this new receiver. When you activate the remote journal later, you can specify that receiver as the starting point.
- Be extra cautious when you have physical file journaling active on the system where the Apply function is running. You may wish to have journaling active here to minimize the time it takes to switch over to the new system. We generally recommend that for the recovery purpose, files on the target should always be journaled. Otherwise, on the target system failure, you need to resynchronize the entire target from the source system.

  If you do have journaling active, journal entries are deposited into the local journal because of the action the Apply function takes. Therefore, ensure that you do not activate the remote journal. If you do, journal entries replicate back to the system that they originated from in the first place.

  Also, you want to ensure that you attach a new receiver onto the local journal when you are switching to the backup system. You need to separate the journal entries, which the Apply process deposits from the journal entries that the applications deposit. The easiest way is to attach a new receiver after the Apply process completes. Journal entries from the application start in this new receiver. Since only the journal entries from the application need to be replicated, you have an identifiable starting point when you activate the remote journal.

Chapter 4. Remote Journal Performance

Performance tests were conducted by the ITSO Rochester to evaluate the following aspects of remote journal support:

- The efficiency of different communications scenarios (OptiConnect, ATM, 100Mbps Ethernet, Token-Ring, and Frame Relay)
- The overall impact of running remote journal in both synchronous and asynchronous modes in an interactive transaction processing environment

Based on the collected benchmark data, the ITSO team made the following conclusions:

- The secret of good journal performance, both local and remote, may be as simple as configuring enough DASD arms and IOPs to service the resulting disk writes.
- If trapped transactions are intolerable in your IT environment, you need to employ synchronous mode. You may consider asynchronous mode if you can tolerate delayed arrival of journal entries (such as in the data warehousing environment).
- Even a high volume of journal traffic generated by a 12-way system running nothing but jobs that produce millions of journal entries cannot saturate OptiConnect and ATM communications hardware.
- When properly-sized communications gear is employed, the performance overhead of the remote journal function is modest.
- Slower communications hardware may be acceptable for environments with fewer transactions and synchronous mode may be appropriate in this environment.

Disclaimer

The following material contains performance values measured in laboratory environments with a specific benchmark. This information is presented along with general recommendations to help you have a better understanding of remote journal function. Your actual performance experience may differ substantially from these values. The benchmark results contained in this chapter have not been submitted to any formal IBM test and are distributed *as is.*

4.1 The Potential Communications Bottleneck

As we approached this project, we expected that communications hardware would become the major bottleneck for fast remote journal performance. Although our benchmark data proved that our concerns were not justified, we believe that you should spend a fair amount of time properly sizing your communications gear.

As with any software that must communicate with other systems, there is a cost associated with the communications layer. This cost can range from extreme, when transmitting over a communications layer lacking sufficient bandwidth (capacity), to minimal when the properly sized communications hardware is used. The communications layer can be divided into two parts:

- Communications software and hardware
- Communications line

The communications software and hardware is what pushes the data down the communications line. For the most part, the difference between the software and hardware is how much data it can push down the communications line at one time. The difference between communication lines is the speed at which data can travel from point A to point B.

Communications hardware bandwidth (capacity) is measured in megabits per second (Mbps). This means how many megabits (millions of bits) can be transferred across the communications hardware per second. This is often referred to as the size of the *communications pipe*. One can visualize this effect by thinking of a pipe with water passing through it. The bigger the pipe is, the larger the volume of water is that can pass through it in any given time.

As far as performance is concerned, the communication pipe is critical because of the impact it can have on remote journal. As explained in this chapter, when using remote journal with undersized communications hardware in either synchronous or asynchronous delivery mode, a noticeable performance impact can be expected. However, when properly sized communications hardware is employed, the performance overhead of remote journal is quite modest.

4.2 Test Environment

Basic performance comparisons were compiled to assist in sizing AS/400 communications hardware. These comparisons can be used for new or existing installations considering the use of remote journal. The purpose of the benchmark was to provide information regarding communications hardware requirements for remote journal. Performance comparisons were gathered for many of today's most commonly used communications hardware.

In the first phase of our tests, we used two 12-way AS/400 systems with Northstar processors. We set up a user ASP with two disk arms and placed the journal receivers there. The disks in the user ASP were unprotected. Our decision to use unprotected disks in this ASP was based on the results of similar performance tests conducted at the Rochester Teraplex Center. The Teraplex team achieved up to twice the throughput of remote journal for mirrored and unprotected environments as opposed to a RAID-5 environment. Our test environment is described in more detail in Table 2.

Table 2. Benchmark Test Systems

System Parameters	Source System	Target System
Model	650 (Northstar)	650 (Northstar)
Processors	12	12
Total Main Storage	11GB	13.5GB
Auxiliary Storage	192GB	94.8GB
Number of Disk Arms in System ASP	52	20
User ASP 2 (receiver ASP)	8.2GB	8.2GB
Number of Disk Arms in ASP 2	2	2
OS/400 Release	V4R3M0	V4R3M0

Since one of the main purposes of our benchmark was to measure the efficiency of different communications scenarios, we set up a range of communications hardware connecting target and source systems. The infrastructure we tested is summarized in Table 3.

Table 3. Benchmark Communications Links

Hardware	Speed	Protocol	Distance between Source and Target
Frame Relay	1.5 Mbps	TCP/IP	20m
Token-Ring	16 Mbps	TCP/IP	20m
Ethernet	100 Mbps	TCP/IP	20m
ATM	155 Mbps	TCP/IP	20m
ATM	155 Mbps	TCP/IP	2km
OptiConnect	1 Gbps		20m
OptiConnect	1 Gbps		500m
OptiConnect	266 Mbps		2km

Additional Benchmark Notes

- The Token-Ring network was not dedicated, which could adversely impact the performance for this protocol.
- The Asynchronous Transfer Mode (ATM) communications hardware had one inline switch. You may experience some performance degradation with a larger number of switches between source and target machines.

4.3 Performance Benchmark Scenario

To benchmark performance, an attempt was made to simulate a "real" user environment with regard to system load. In this benchmark scenario, a CPU load of 70% or higher was maintained. DASD utilization was at 4.9% during testing. Since the purpose of these benchmarks was to compare communications hardware, not DASD utilization, DASD was purposely configured at high capacity to ensure that it would not be a bottleneck. We initially started our tests on a AS/400 system with an impressive configuration. It included 11GB of main memory and over 180 disk arms with a total capacity of more than 700GB. However, we quickly realized that this environment would not be representative for the majority of existing AS/400 customers and decided to create a *constrained hardware configuration*. We removed over 120 disk arms from the source machine configuration. All disk drives in the source machine had the same capacity of 4GB. To constrain the main memory available to benchmark jobs, we created a dedicated batch subsystem and assigned it to a shared memory pool with 1.5GB of main memory.

The programs that were developed to simulate a system load submitted 500 jobs. Each job ran 15 times (iterations), and each iteration generated 250 database transactions. The transactions were a mixture of database inserts, updates, and deletes. All benchmark tests were run with commitment control. To achieve high CPU utilization, we decided to eliminate any "think time", which means that we generated a steady aggressive workload.

All 500 jobs were submitted to the batch subsystem. The submitted jobs ran in a shared pool, which had a size of 1.5GB. This translates into 3MB of main memory for each active job. Prior to each run, the shared pool was cleared with the CLRPOOL command to remove all cached data from the main memory. The job queue associated with this subsystem was initially held. All jobs were submitted, and the job queue was released in hopes that this would ensure that all jobs start and end at the same time. All subsequent benchmark testing was performed in the same manner.

Table 4. Benchmark Transactions

Number of Jobs	Number of Iterations	Number of Transactions	Files Used	Avg. Journal Entries Generated	Size of Generated Receiver
500	15	250	2	4571000	1623MB

One of the goals for our test scenario was to avoid having the physical files linger in main memory. We believed that active paging better reflects real life environments. We ran tests on a large system with an ample amount of main memory. To make sure that data was actually read from I/O rather than the main store, we pre-populated both physical files. Their total size substantially overcommitted main memory. No substantial portion of either physical file could remain resident, but instead had to be frequently re-faulted during the run.

Table 5. Physical Files Used in Tests

File Name	File Size	Number of Records
ITEM_FACT	90GB	270000000
CUST_DIM	1.35GB	6750000

This strategy proved to be effective. We used performance tools to measure DASD I/O activity. We discovered that before we created the large files, the DASD I/O was minimal since the files were pre-loaded into main memory. Once we created the larger files described in Table 5, we observed a consistent level of page faults from DASD.

The following steps describe the test scenario we used for different communications links and delivery modes:

1. Inactivate the remote journal on the source system.
2. Remove the remote journal association on the source system
3. Delete the remote journal and receivers on the target system.
4. Change the local journal, and generate a new receiver on the source system.
5. Add a remote journal on the source system.
6. Activate the remote journal. Depending on the test scenario, we activated the remote journal either in synchronous or asynchronous mode.
7. Run the benchmark. All the tests were run with commitment control.

Please refer to Appendix A, “Benchmark Scenario” on page 101, for a detailed description of the benchmark steps.

4.4 Remote Journal Performance Benchmarks

Many horror stories abound regarding poorly configured systems on which no journaling overhead can be tolerated. We discovered that even high volume journal environments can be handled with little, if any, discernible extra performance overhead provided a sufficient quantity of fast DASD arms and IOPs are configured. We witnessed very little difference in the benchmark run with and without journaling enabled that we elected to avoid producing a non-journaling versus a journaling baseline run. If your experience with journaling shows higher quantities of overhead, you may want to examine your underlying DASD and IOP configuration. Make sure that you adequately size this subsystem to support the synchronous write traffic that journaling produces.

Our experience demonstrates that the secret of good journal performance may be as simple as configuring enough DASD arms and IOPs to service the resulting synchronous disk writes. This is especially true if the resulting DASD traffic is over committing the fast write cache of your IOP. It often makes sense to assure that each of the disk arms residing within the ASP housing your journal receiver are attached to a different IOP.

4.4.1 Local Journal

The first benchmark was conducted using conventional local journaling to obtain a baseline for subsequent results.

Table 6. Local Journal

Hardware Configuration	Elapsed Time (sec)	AVG CPU Time (sec)	CPU Utilization	Transaction Rate (sec)	Transport Rate (B/s)
Local Journal	1417	679	83.95	2648.17	N/A

Benchmark Table Legend

Hardware Configuration Communications hardware tested.
Elapsed Time (sec) Longest of 500 jobs.
AVG CPU Time (sec) CPU time of 500 jobs divided by 12 processors.
CPU Utilization Performance Monitor data; first and last interval dropped.
Transaction Rate (sec) 3750000 transactions divided by Elapsed Time. The following formula was used to calculate the transactions number:
Transactions = Number of Jobs x Number of Iterations x Number of Transactions per Iteration x Number of Files
Transport Rate (B/sec) Bytes added to the remote receiver divided by Elapsed Time.

In our test environments, we noticed the throughput for the local journal run was roughly equivalent to some of the remote journal runs due to the following changes. As of V4R2, journal deposit bundling balances response time versus CPU utilization. Additionally, when remote journals are associated with a local journal, the bundling characteristics are slightly modified to extend the wait time per journal bundle. The goal is to send larger bundles across the communications line and IOP to minimize the CPU utilization impact of the communications request.

4.4.2 Synchronous Remote Journal

Table 7 shows the benchmark results for remote journal using synchronous delivery mode. The comparison is made over several hardware configurations to facilitate the process of selecting such a communications hardware.

Table 7. Synchronous Remote Journal

Hardware Configuration	Elapsed Time (sec)	Avg. CPU Time (sec)	CPU Utilization	Transaction Rate (sec)	Transport Rate (B/s)
Frame Relay 1.5 Mbps TCP/IP	8355	556	11.93	448.83	194255
Token Ring 16 Mbps TCP/IP	3893	564	26.54	963.27	416902
Ethernet 100 Mbps TCP/IP	1673	638	66.60	2241.48	970114
ATM 20m 155 Mbps TCP/IP	1385	650	82.85	2707.58	1171841
ATM 2km 155 Mbps TCP/IP	1425	647	79.79	2631.58	1138947
OptiConnect 20m 1 Gbps	1392	661	83.91	2693.97	1165948
OptiConnect 500m 1 Gbps	1409	657	83.32	2661.46	1151881
OptiConnect 2km 266 Mbps	1409	666	81.24	2661.46	1151881

The results of the OptiConnect and ATM hardware configurations are within acceptable ranges when compared to the local journal baseline. However, a significant dropoff emerges with the use of 100Mb Ethernet, Token-Ring, and Frame Relay hardware configurations. Notice the dropoff in CPU Utilization and Time, and the increase in Elapsed Time. These changes indicate that the system (CPU) is waiting on the smaller capacity communication hardware since the CPU is not fully utilized.

Based on the observed benchmark information, we conclude that such a high volume of journal traffic (as produced by a screaming 12-way that is running nothing but jobs producing millions of journal entries) can simply overcommit an undersized set of communications hardware. A customer with a different mix of application programs running on the same 12-way and slower communications hardware may find synchronous mode quite acceptable. The reason is simply because their resulting volume of generated journal entries are more modest than what our particular benchmark produced.

Note that, with the load, we were able to generate on the 12-way systems. We did not saturate OptiConnect and ATM communications hardware. Therefore, both are a valid choice for an environment similar to the one we were testing. Note also that increasing the distance between source and target machines by switching in additional optical fiber did not significantly increase the elapsed time.

If trapped transactions are intolerable in your IT environment, you need to employ the synchronous mode, regardless of the cost or overhead. You need to size the communications hardware properly to assure that you can safely use the synchronous mode without suffering much performance overhead. ATM and OptiConnect are good choices even for environments that experience a high volume of journal traffic. If your volume of journal traffic is more modest, you may find that you can comfortably elect to employ synchronous mode and enjoy its benefits without stepping up to top-of-the-line communications gear.

As the data demonstrates, synchronous mode is a great choice that consumes hardly any noticeable overhead, provided that you sized the communications hardware adequately.

4.4.3 Asynchronous Remote Journal

The asynchronous alternative for remote journaling offers a choice for those who intend to implement remote journaling as part of their solution, but:

- Do not want to upgrade their communications hardware
- Do not need the timeliness or overhead of synchronous remote journaling

You may consider the asynchronous mode if your shop can tolerate the delayed arrival of journal entries. It may be intolerable to switch over and have your business move slowly along on the target machine with a few of the final database transactions still trapped on the source machine because these transactions were not sent to the target in realtime. In this case, you need to give up on asynchronous and move back to synchronous mode.

Please note that with the asynchronous delivery mode, both elapsed and CPU time fell to levels of local journal benchmarks without a noticeable CPU utilization increase. Refer to Table 8 for the test results.

A second elapsed time was added to this table. This number (the bottom number) represents the delay in arrival of the journal entries to the target system. It is important to understand that the use of less efficient communication hardware means that more journal entries may not reach the target system if the source system fails.

Table 8. Asynchronous Remote Journal

Hardware Configuration	Elapsed Time (sec)	CPU Time (sec)	CPU Utilization	Transaction Rate (sec)	Transport Rate (B/s)
Frame Relay 1.5 Mbps TCP/IP	1512 lcl 3886 rmt	637	73.65	2480.16	417653
Ethernet 100 Mbps TCP/IP	1388 lcl 1388 rmt	645	83.87	2701.73	1169308
ATM 2km 155 Mbps TCP/IP	1386 lcl 1403 rmt	648	83.67	2705.63	1156807
OptiConnect 500m 1 Gbps	1386 lcl 1389 rmt	648	83.56	2707.58	1168467
OptiConnect 2km 1 Gbps	1390 lcl 1390 rmt	652	83.58	2697.84	1167626

Table 8 indicates that with efficient communications hardware, in most cases, asynchronous remote journal is maintained almost in realtime on the target system. The table also indicates that even when configured with a 1.5 Mbps Frame-Relay connection, the source system shows only minor performance impact. There is, however, a noticeable delay in the journal entries arriving on the target system. However, there is the risk of having recent transactions trapped on the production machine if a Frame Relay connection is used.

4.4.4 Remote Journal Catch-Up Mode

The remote journal catch-up mode is independent of the selected delivery mode. It constitutes the most efficient way to transport journal entries between source and target systems. In the catch-up mode, the data is sent to the target system in large blocks until the target system journal is re-calibrated to the source system journal. When complete, the system automatically switches to continuous mode.

For this comparison, the benchmark job statistics were not required or recorded. The pertinent data is the time required to populate the target receiver and the transport rate of each hardware configuration. The Elapsed Time represents the duration of time required to populate the target journal with the contents of the source journal. The Transport Rate represents the rate of data transferred per second shown in kilobytes.

For these tests, we used a journal receiver created during our synchronous mode benchmark. This journal receiver contained 4571318 journal entries and had a size of 1.623GB. The following steps illustrate the test scenario we used for the catch-up mode:

1. Inactivate the remote journal on the source system.
2. Delete the remote journal and receivers on the target system.
3. Add a remote journal on the source system.
4. Activate the remote journal. This initiates the catch-up mode. Start the Elapsed Time measurement.
5. Wait until the last journal entry from the source receiver arrives at the target receiver. Stop the Elapsed Time measurement.

All tests were run on the 12-way systems described in Table 2 on page 70. The point of having sufficient communications hardware is amplified in Table 9.

Table 9. Remote Journal Catch-up Mode

Hardware Configuration	Elapsed Time (sec)	Transport Rate (B/s)
Token-Ring 16 Mbps TCP/IP	2733	582656
Ethernet 100 Mbps TCP/IP	327	4980736
ATM 2km 155Mbps	238	6999040
OptiConnect 20m 1 Gbps	150	11079680

OptiConnect, when compared to Token-Ring data in this benchmark, is approximately 19 times faster in transmission speed (11079680 / 582656 =

19.02). Table 9 shows that OptiConnect was approximately 1.5 times more efficient than ATM (11079680 / 6999040 = 1.58). We did not observe any measurable impact of the catch-up mode on the CPU utilization when benchmarking the 12-way systems.

The true importance of the catch-up mode becomes evident on the far side of a crash of your production system when you need to re-calibrate the two machines and transport everything that happened on the target machine back to the source machine. For example, imagine that your application generates 8GB of journal receiver content per eight-hour shift. Your machine crashes at 6 a.m. You have been running for eight hours on the backup system and now you want to switch back to the production system. The pertinent question is: How long will it take to ship eight hours worth of production data, 8GB in this case, back to the production machine? The answer is: If you run over OptiConnect, it takes only 12 minutes and 55 seconds. Of course, you still have to apply all those changes before you can switch back to the production system.

4.5 Additional Benchmark Data

Benchmark testing was also performed on additional AS/400 systems with lower capacity and limited communications hardware for further analysis. The purpose of this benchmark was to compare the efficiency of different communications gear. Therefore, we do not provide results for the local journal environment.

Table 10. Additional Benchmark Systems

System Parameters	Source System	Target System
Model	170	170
Processors	#2292	#2292
Total Main Storage	1024MB	1024MB
Auxiliary Storage	32GB	16GB
Number of Disk Arms in System ASP	7	3
User ASP 2 (receiver ASP)	4GB	4GB
Number of Disk Arms in ASP2	1	1
OS/400 Release	V4R3M0	V4R3M0

The test scenario required modification due to the lower capacity of the systems. In the modified test scenario, instead of submitting 500 jobs with 15 iterations each, 125 jobs with 20 iterations were submitted. Table 11 shows the modified benchmark scenario for 170 systems.

Table 11. Modified Benchmark Scenario

Number of Jobs	Number of Iterations	Number of Transactions	Files Used	AVG Journal Entries Generated	Data Generated
125	20	250	2	1516200	496MB

The two communications hardware options tested on these machines were Ethernet at 100 Mbps and ATM at 155 Mbps.

Please note how the different testing environment affected the elapsed time and transport rate. Running at nearly 100% CPU utilization, the 170 system was not able to generate enough data base changes and journal entries to saturate Ethernet or ATM. In both synchronous and asynchronous modes, the performance of these two communications options was similar. That was not the case when we ran our benchmark on the 12-way 650 models. Here, ATM allowed a 20% greater throughput than a 100Mb Ethernet in synchronous mode. Refer to Table 7 on page 74.

The following two tables summarize our 170 system benchmarks.

Table 12. Synchronous Remote Journal—Model 170 Benchmark

Hardware Configuration	Elapsed Time (sec)	AVG CPU Time (sec)	CPU Utilization	Transaction Rate (sec)	Transport Rate B/sec
Ethernet 100 Mbps TCP/IP	1525	1065	98.63	820	341139
ATM 20m 155 Mbps TCP/IP	1487	1055	99.85	841	349856

Table 13. Asynchronous Remote Journal—Model 170 Benchmark

Hardware Configuration	Elapsed Time (sec)	AVG CPU Time (sec)	CPU Utilization	Transaction Rate (sec)	Transport Rate B/sec
Ethernet 100 Mbps TCP/IP	1478 lcl 1478 rmt	1042	99.99	846	351987
ATM 2km 155 Mbps TCP/IP	1459 lcl 1459 rmt	1035	99.99	857	356571

Note: For the asynchronous mode test the difference between the elapsed time on the remote and local systems was consistently below 0.5 sec.

After running our benchmark on the 170 systems, we concluded that slower communications hardware may be acceptable for a slower CPU. Even synchronous mode may be quite appropriate in this environment.

For the catch-up mode tests, we used journal receivers created specially for this benchmark with the size similar to the receiver size for the 12-way systems benchmark. The reason for doing so was that we wanted to avoid a potential influence of the initial catch-up mode "warm-up" time on the benchmark results. The journal receiver for the Ethernet test contained 4547956 journal entries and had a size of 1456MB. The receiver for the ATM test contained 4548657 entries and had a size of 1464MB.

Table 14. Catch-up Mode—Model 170 Benchmark

Hardware Configuration	Elapsed Time (sec)	Transport Rate (B/s)
Ethernet 100 Mbps TCP/IP	668	2286322
ATM 2km 155 Mbps TCP/IP	259	5928635

In a catch-up mode, the slower CPU and overall system configuration had a negative impact on the transport rate. Even running at full speed of the communication links, we only achieved 45% of transport rate of a 12-way system for the Ethernet and 84% for the ATM. Please compare data in Transport Rate column in Table 9 on page 76 to the data in Table 14.

For the 170 models, we also observed that running in catch-up mode without any additional system activity added 6% to the CPU utilization on both the source and target systems.

4.6 Performance Recommendations

There are several factors to consider when tuning remote journal performance.
Note: Some of these improve local journal performance as well.

- Make sure you properly size the communications hardware.
- Consider creating a user ASP on the target system for the remote receivers, and mirror the ASP at the disk level.
- Assure that the ASP housing the target receiver has at least as many disk arms as the ASP of the matching receiver from the source system.
- Remove internal journal entries using the RCVSIZOPT(*RMVINTENT) parameter of the CHGJRN command. This prevents sending access path journal images to the target system.
- Consider reducing the width of journal entries using the RCVSIZOPT(*MINFIXLEN) parameter of the CHGJRN command. This minimizes the number of bytes per journal entry sent to the target system and to the disk.
- Consider reducing the number of journal entries by using the OMTJRNE(*OPNCLO) parameter of the STRJRNPF command. This prevents open and close operations on the specified files from generating open and close journal entries.

Chapter 5. Building the Switchover Solution

This chapter covers the use of remote journal to create a high availability environment. It also discusses other factors that impact the creation of a high availability solution. These factors are not trivial. The user must give careful consideration about whether the business has adequate programming resources to construct a solution itself or instead consider one of the third-party high availability applications available for the AS/400 system.

Note

References to a "high availability application" in the following text presume one of the third-party high availability applications available for the AS/400 system.

5.1 Initial Setup Tasks for the Environment

For example purposes, this chapter uses a simple application with a single database and journal, as illustrated in Figure 56. This example is not meant to imply any limitation or restriction to remote journal with regard to the number of files that can be handled by remote journal. Nor does it imply limitation or restriction to the number of remote journals that can be established between a pair of systems. Normal business work occurs on the production system, which is named SRCSYS in this example. The backup system contains the replica database and is named TGTSYS.

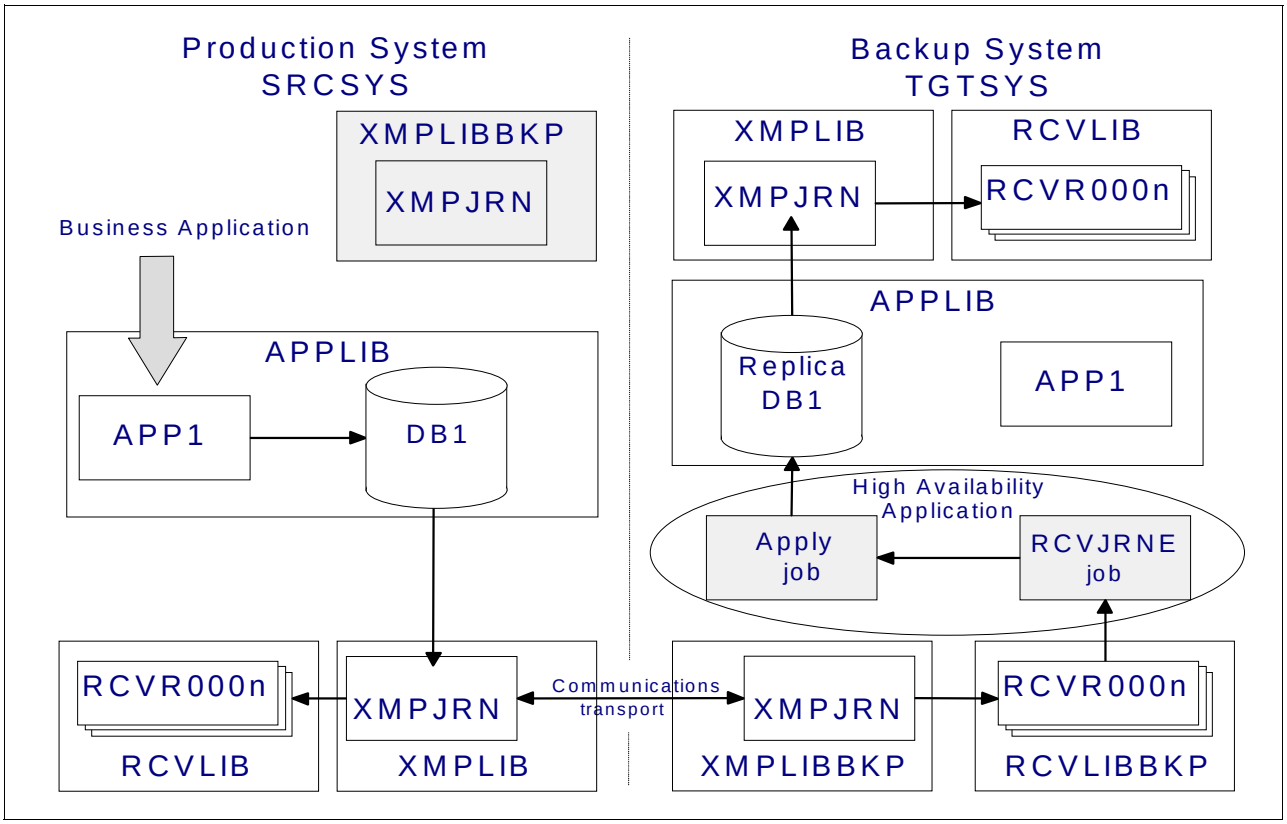


Figure 56. Example Application

5.1.1 The Production Database and its Journals

The example application, APP1, operates against database DB1 on SRCSYS. Both the application and the database reside in APPLIB. DB1 is journaled to XMPJRN in XMPLIB. The associated journal receivers are stored in RCVLIB, and the current chain starts with receiver RCVR0001. As APP1 makes updates to DB1, the entries are recorded in the receivers attached to XMPJRN.

5.1.1.1 The Remote Journal on the Backup System

XMPJRN on SRCSYS has an associated remote journal of the same name on TGTSYS. However, the journal is redirected to a different library, XMPLIBBKP. A *TYPE1 remote journal is specified, and therefore, receivers saved from the remote journal receiver chain can be restored to the local journal receiver chain. The remote journal receivers are stored in RCVLIBBKP with the same names as on SRCSYS. Entries written to the receivers on SRCSYS are also written to the remote journal receivers on TGTSYS.

5.1.2 The Replica Database and its Journals

A copy of DB1 (made using Save and Restore) exists on the backup system, TGTSYS. The backup DB1 is also journaled to XMPJRN in XMPLIB on TGTSYS. As the high availability application's apply process updates DB1 on TGTSYS, the entries are recorded in the receivers attached to XMPJRN.

5.1.2.1 The Remote Journal on the Production System

The remote journal associated with XMPJRN on TGTSYS is stored in library XMPLIBBKP on SRCSYS, but it is inactive. This reduces excess and unnecessary communication traffic. The remote journal is activated only in the case of a switchover, when it is used to replicate data from TGTSYS back to SRCSYS.

5.2 Activating the Switchover Operation

For establishing the example environment, APP1 and DB1 are assumed to already exist on the production system, with DB1 journaled to XMPJRN. If an application does not already use journaling, a journaling environment will be required. The high availability application generally creates this automatically.

To establish a journaling environment manually, create the associated journal receiver library, journal receiver, journal library, and journal (the journal receiver must exist before creating the journal). Start journaling against the database. Figure 57 illustrates the commands for the example application.

```
CRTLIB LIB(RCVLIB)
CRTJRNRCV JRNRCV(RCVLIB/RCVR0001)
CRTLIB LIB(XMPLIB)
CRTJRN JRN(XMPLIB/XMPJRN) JRNRCV(RCVLIB/RCVR0001) +
       RCVSIZOPT(*RMVINTENT *MINFIXLEN)
STRJRNPF FILE(APPLIB/DB1) JRN(XMPLIB/XMPJRN) OMTJRNE(*OPNCLO)
```

Figure 57. Creating the Journaling Environment

Establishing the backup system configuration requires a brief period of downtime for the application. Activity against the database must be stopped to establish a guaranteed synchronization point between the two systems. When there is no

activity against the database, use the high availability application to create a duplicate of the application and the database on the backup system.

Alternatively, save the application and database from the production system and restore it to the backup system. This can be done either using a communications link or removable media. Figure 58 and Figure 59 illustrate the commands for the example application.

```
SAVRSTLIB LIB(APPLIB) RMTLOCNAME(TGTSYS)
```

Figure 58. Creating the Backup Application Using Communications

```
On SRCSYS:
SAVLIB LIB(APPLIB) DEV(device-name)

On TGTSYS:
RSTLIB SAVLIB(APPLIB) DEV(device-name)
```

Figure 59. Creating the Backup Application Using Removable Media

Immediately after the file is saved on the production system, change the journal and attach a new journal receiver on the production system. This provides an exact reference point where new transactions begin following the initial synchronization point. Figure 60 illustrates the command for the example application.

```
CHGJRN JRN(XMPLIB/XMPJRN) JRNRCV(*GEN)
```

Figure 60. Changing the Journal

Assuming that the previous receiver was still the original receiver, RCVR0001, this creates RCVR0002. Normal business activity using the application on the production system can resume after changing the journal receiver.

After restoring the application and database to the backup system, perform the same steps to create and start the journaling environment for the backup database. Although this journal is not needed until a switchover is performed, we strongly recommend that the files on the backup are always journaled. Otherwise, when the backup fails, you need to resynchronize the entire backup system from the production machine.

5.2.1 Creating the Remote Journal Environment

Use the high availability application to establish the remote journal environment on the backup system. Alternatively, create the remote journal and journal receiver libraries on the backup system. These must exist before the remote journal can be created. Figure 61 illustrates the command for the example application.

```
CRTLIB LIB(RCVLIBBKP)
CRTLIB LIB(XMPLIBBKP)
```

Figure 61. Creating the Remote Journal and Journal Receiver Libraries

Use the Add Remote Journal (ADDRMTJRN) command or the Add Remote Journal (QjoAddRemoteJournal) API to associate a remote journal on the backup system with the journal on the production system. Specify the remote journal and journal receiver libraries that you just created. The add process creates the remote journal and sets it to an *INACTIVE journal state. Figure 62 illustrates the command for the example application.

```
ADDRMTJRN RDB(TGTSYS) SRCJRN(XMPLIB/XMPJRN) TGTJRN(XMPLIBBKP/XMPJRN)
          RMTRCVLIB(RCVLIBBKP)
```

Figure 62. Adding the Remote Journal

Perform the same steps against the journal on the backup system to establish the remote journal environment back to the production system. The remote journal on the production system is left in the *INACTIVE journal state until a switchover is performed.

5.2.2 Catch-Up and Continuous Modes

Remote journal can now be activated and entries applied to the backup database. On the production system, use the Change Remote Journal (CHGRMTJRN) command or the Change Journal State (QjoChangeJournalState) API to change the remote journal state to *ACTIVE. Start with the first receiver immediately following the initial synchronization point. Figure 63 illustrates the command for the example application.

```
CHGRMTJRN    RDB(TGTSYS) SRCJRN(XMPLIB/XMPJRN) TGTJRN(XMPLIBBKP/XMPJRN)
             JRNSTATE(*ACTIVE) DELIVERY(*SYNC)
             STRJRNRCV(RCVLIB/RCVR0002)
```

Figure 63. Starting Remote Journal

This creates and attaches the first receiver to the remote journal, in this case RCVR0002. Remote journal goes into the catch-up mode. In this mode, it transmits the entries that are in the receivers in the local journal receiver chain as quickly and efficiently as possible to the remote journal receivers on the backup system. The high availability application apply process should also be started to read the remote journal entries and update the backup database accordingly.

When all of the entries that existed in the local journal receivers are transmitted to the remote journal, remote journal switches to the continuous mode. As entries are added to the local journal receivers, they are transmitted to the remote journal using the delivery method specified when remote journal was activated. The high availability application apply process reads the new entries as they are added and maintains the synchronization of the production and backup databases.

5.3 Failure Recovery and Switchover to the Backup System

In the event of an unplanned outage, the backup system must be prepared so that it can assume the role of the production system. This process involves executing a number of steps in a specific order. These steps can be done manually. However, the needs of the business often dictate the use of an automated switching solution for speed and to minimize errors. Most high

availability applications include automated switching solutions. These products handle most, if not all, of the common steps and allow customization for business-specific requirements.

5.3.1 Preparing the Backup System

During normal operations, the remote journal function replicates journal entries to the backup system, which the high availability application retrieves from the journal and applies to the backup database. When the production system experiences an unplanned outage, this may result in journal entries that arrived in the remote journal but were not applied yet. The availability application should complete the application of these entries before a switch is performed (see Figure 64).

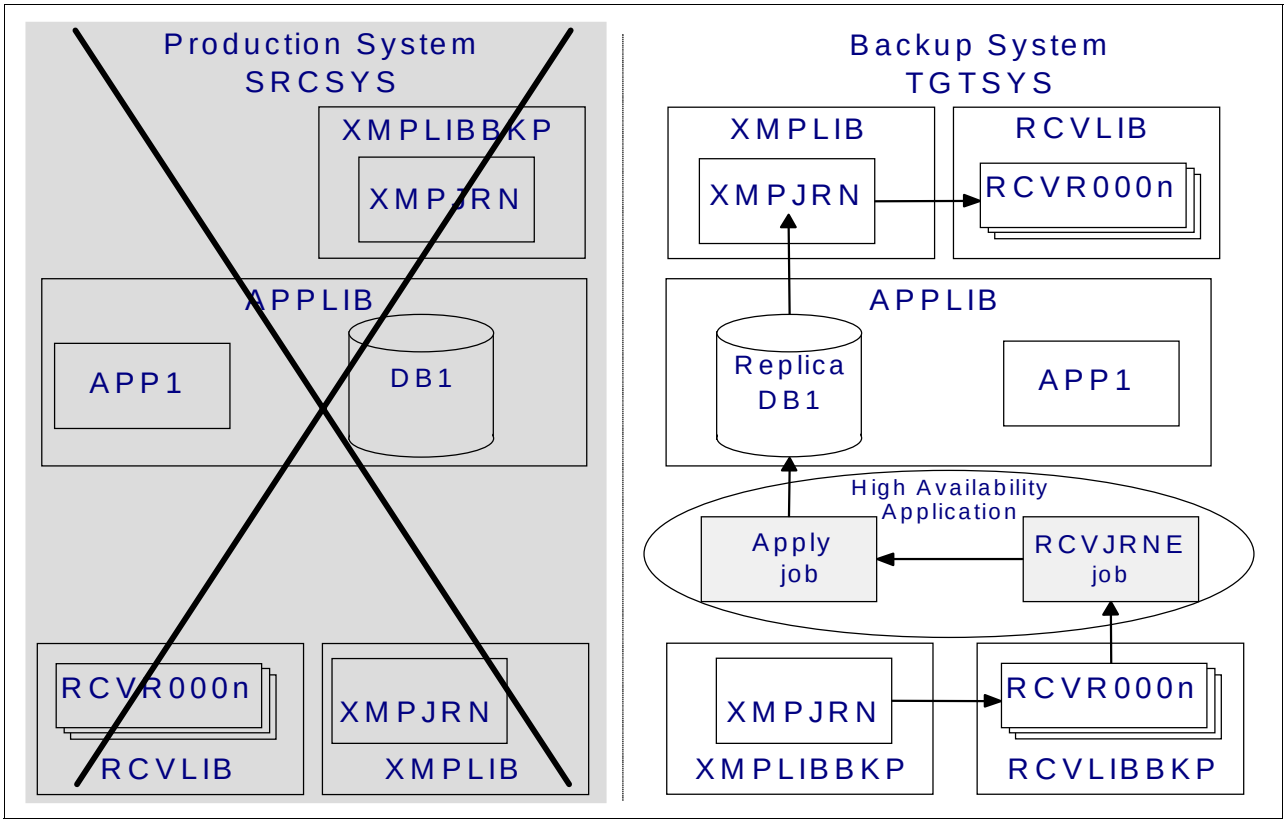


Figure 64. Completing the Apply

If there are any open commit cycles after all the received journal entries are applied, these should be rolled back to insure that there are no partial transactions recorded. The rollback should be done immediately after all the entries are applied, unless synchronous remote journal is used. For synchronous remote journal, the rollback may be delayed depending on the disposition of unconfirmed journal entries (see 5.3.2, “Replaying the Unconfirmed Journal Entries” on page 86).

When the backup database contains all of the completed entries received from the production system, deactivate remote journal on the backup system. Use the Change Journal (CHGJRN) command or the Change Journal State (QjoChangeJournalState) API to change the remote journal state to *INACTIVE. This removes any unconfirmed journal entries from the remote journal receiver

and prevents unexpected conditions when the production system comes back online. Figure 65 illustrates the command for the example application.

```
CHGJRN JRN(XMPLIBBKP/XMPJRN) JRNSTATE(*INACTIVE)
```

Figure 65. Deactivating a Remote Journal

As previously recommended, journaling should already be started for the backup database. If journaling is not already started for the backup database, start it at this point.

Note

Starting journaling at the time of a switchover can significantly extend the amount of time for the switchover if there are a number of files to be journaled.

After all of the entries are applied to the backup database, the journal receiver for the local journal should be changed. This provides an exact reference point for where transactions from the production system ended and transactions generated on the backup system begin. This reference point is important for when the production system is returned to an operational status. Activity that occurs on the backup system is replicated back to the production prior to the production system being made available for normal use again. If the database does not need to be synchronized using save and restore (see 5.5.2, “Synchronizing after a Switchover” on page 90), the journal entries that occur after the reference point may be used to synchronize the database.

Other considerations, such as user access to the backup system and business and application-specific requirements, also affect the preparation of the backup system. The switchover process must address these items as well (see 5.6, “Other Considerations for Switchover” on page 91).

Any changes made to the database on the backup system are recorded in the local journal (XMPJRN in XMPLIB) on TGTSYS. When the production system (SRCSYS) returns to use, these entries are transferred to SRCSYS to be applied to the production database. From the perspective of the high availability application, the flow of data has switched, now running from TGTSYS to SRCSYS (see Figure 67 on page 89).

5.3.2 Replaying the Unconfirmed Journal Entries

When using synchronous remote journal, some entries may reside in the remote journal receiver in an unconfirmed state. The operating system writes entries to the remote journal receiver first, then to the local journal receiver, and finally to the file. When the operating system writes the entry in the local journal receiver to secondary storage, the entry is marked as confirmed in the remote journal.

Unconfirmed entries are not usually retrieved from a remote journal receiver. The Receive Journal Entry (RCVJRNE) command defaults to include only confirmed entries from the remote journal receiver (INCENT(*CONFIRMED)). To receive unconfirmed entries, INCENT(*ALL) must be explicitly specified.

After applying all confirmed journal entries (see 5.3.1, “Preparing the Backup System” on page 85), any unconfirmed changes can also be applied. The receiver change on the local journal should be done prior to applying any unconfirmed entries, so these entries are captured as an activity on the backup system. Because they are unconfirmed, the entries may or may not be written to the production database. They need to be replicated back to the production system and applied along with all other activity that occurs on the backup system during the outage. After all unconfirmed changes are applied to the database, any commit cycles that are still open should be rolled back to insure that there are no partial transactions recorded.

5.3.3 Restarting the Application on the Backup System

After the backup system is prepared, users may be allowed onto the backup system, and the application can be restarted on the backup system. If feasible, users should attempt to verify that the last transaction they were working on completed successfully. If not, the transaction should be repeated.

Business activity can now resume on the backup system (within the constraints of the company’s downtime contingency plan). The local journal records all of the activity that occurs on the backup system (see Figure 66). The journal receivers associated with that journal must be retained until the production system is returned to service and all the entries are transmitted and applied to the production database.

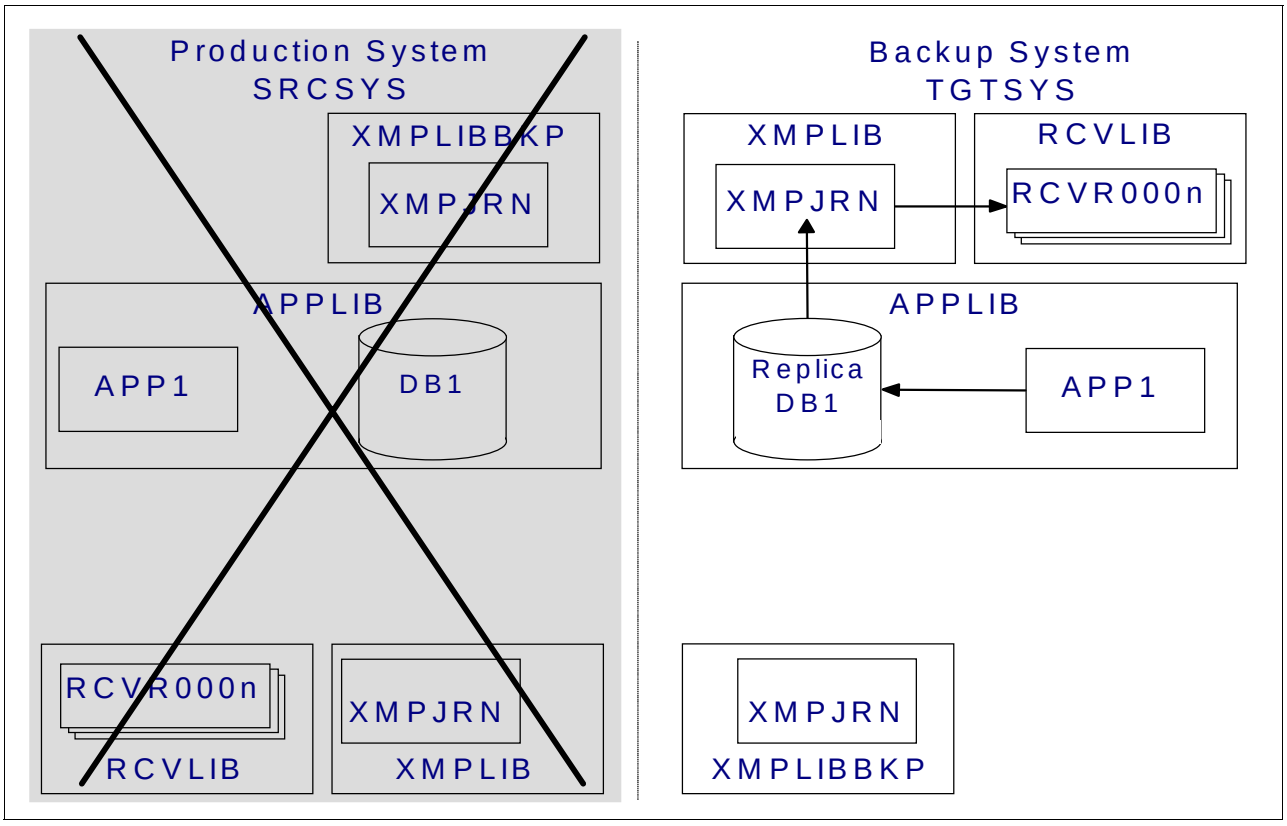


Figure 66. Resuming Work on the Backup System

5.4 The Switchback to the Production System

When the production system returns to service, it must be synchronized with the backup before normal activity can resume. The production database must be updated with the activity that occurred on the backup during the outage. Other critical objects that changed during the outage may also need synchronization (see 5.6, “Other Considerations for Switchover” on page 91).

5.4.1 Resynchronizing the Production and Replica Databases

As soon as the production system returns make sure that remote journal is not activated on the production database’s local journal. This prevents any unexpected and unnecessary data flow while synchronizing the production database.

5.4.1.1 Recovery with Asynchronous Remote Journal

When using asynchronous remote journal, some entries may not have been replicated to the backup system prior to the outage. Compare the local journal receiver on the production system and the remote journal receiver on the backup system to determine the number and type of entries that were not replicated. Depending on the outcome of the analysis, the user has two options:

- The entries that were not replicated are entered into the backup database, either through the availability application or by "patching" the database through such means as SQL, the DFU editor, or using the Copy File (CPYF) command. Take care not to invalidate any transactions that occurred on the backup during the outage.
- The entries are ignored and the backup database in its current state is considered to be up-to-date.

If the second option is selected, or if the relative record positioning of the file records was not preserved during the update of the backup database, it may be necessary to save the backup database and restore it to the production system. This forced synchronization can be time consuming and may be undesirable in some business situations. See 5.5, “Save and Restore Scenarios” on page 90 for additional details.

5.4.1.2 Recovery with Synchronous Remote Journal

When using synchronous remote journal (or when relative record positioning is maintained between the files), it is usually possible to use the availability application to synchronize the files while continuing normal business operations. As previously recommended, a remote journal associated with the local journal on the backup system should already exist on the production system, in a deactivated state. If it does not, create the remote journal prior to beginning the synchronization process.

Activate remote journal from the backup to the production system. The starting journal receiver should be the first receiver in the chain that records the activity on the backup system after the switchover. Remote journal goes into the catch-up mode. It sends the journal entries from the backup system to the production system as quickly as possible until the two journals are synchronized.

Start the availability application on the production system to apply the journal entries from the remote journal to the production database. This should include

any unconfirmed entries that were applied to the backup database at the start of the switchover and all activity after that (see Figure 67). Please note that some of the unconfirmed entries may have been applied on the production system before the failure have occurred. The high availability application should know how to handle those cases and avoid "re-doing" the work.

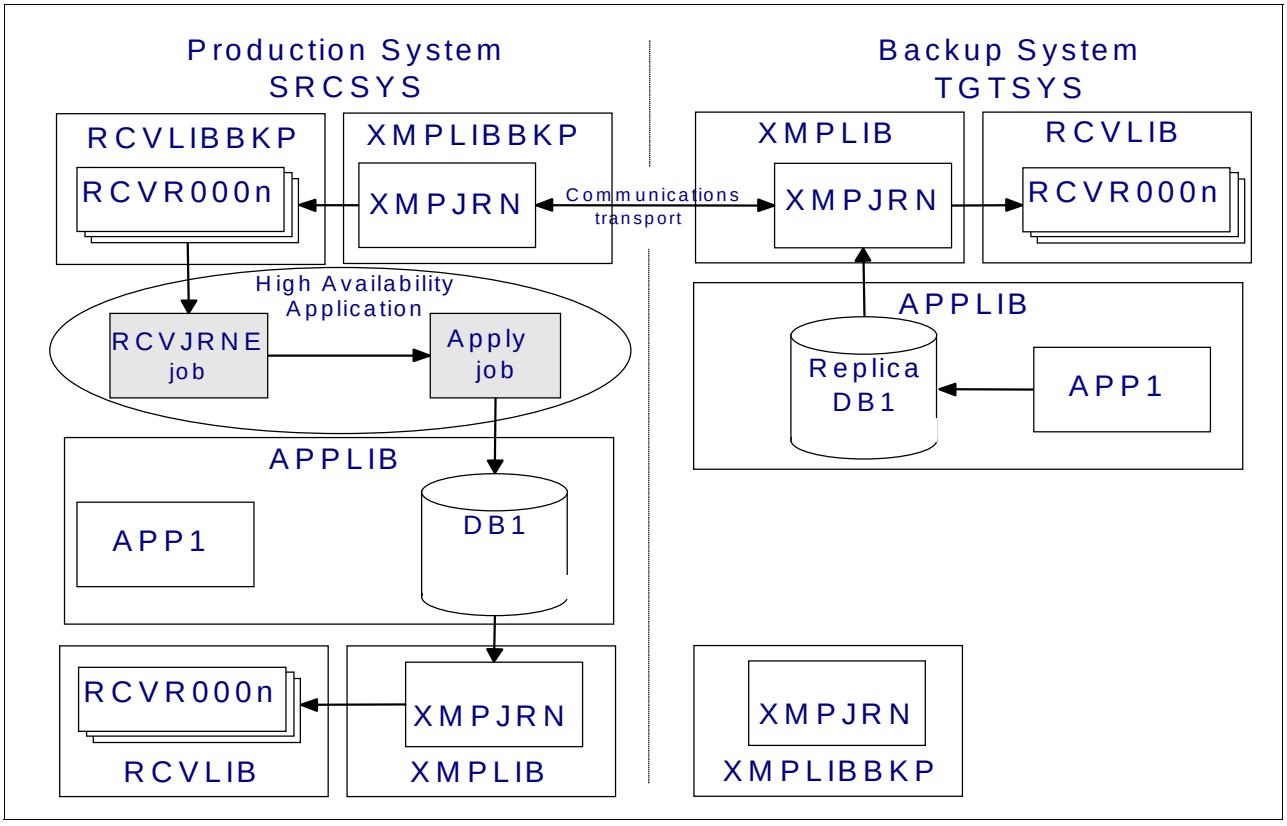


Figure 67. Resynchronizing the Production System

After the remote journal catches up with the backlog of journal entries, it switches automatically into continuous mode. Normal business activity can continue on the backup system until a convenient time to switch users back to the production system. To minimize the impact of the switchback, allow the high availability application to complete processing on the backlog of journal entries from the remote journal on the production system and reach a state where it is current with the backup system.

5.4.2 The Switchback Process

When the production system is prepared to resume normal operations again, normal business activity should be temporarily suspended. Allow the availability application to apply all of the entries from the remote journal until the two databases are completely synchronized. Complete the following steps to switch back to the production system.

1. Inactivate remote journal from the backup system to the production system and end the availability application.
2. Perform any switching activity required to allow users to access to the production system again, and re-start the high availability application.

3. Change the local journal to attach a new journal receiver for the local journal on the production system. This provides a starting point for remote journal and prevents the remote journal from going into an unnecessary catch-up mode.
4. Activate the remote journal from the production system to the backup system. Specify the new receiver as the starting point.
5. Start the availability application on the backup system to begin applying new entries.
6. Allow users back onto the production system, and resume normal business activity.

5.5 Save and Restore Scenarios

Remote journal in a high availability environment can simplify normal backup strategies. High availability applications may also present situations where the user may be forced to save and restore the database. These circumstances vary based on business needs and requirements for availability.

5.5.1 Backing Up Files and Journal Receivers

Even when maintaining a second system as a hot backup, most users still want to back up the database as part of a normal recovery strategy. This includes regularly saving the entire database on a periodic basis (weekly, biweekly, monthly, and so on) and saving the journal receivers associated with the database journal on a more frequent basis, usually daily.

With remote journal, all of this activity can be done on the backup system without affecting normal business operations. At a regularly scheduled time, change the receiver on the production journal. This also changes the receiver on the remote journal. Then, save the detached remote journal receivers on the backup system. If the remote journal is a *TYPE1 remote journal, the receivers are identical to the receivers on the production system. These receivers can be restored to the receiver chain on either system.

To save the database, end the high availability application apply process. This provides a "point-in-time" snapshot of the database without affecting normal business operations on the production system or the continued transmission of journal entries to the backup system. After saving the database, start the high availability application apply process to process the journal receiver entries that built up during the save process.

5.5.2 Synchronizing after a Switchover

During a switchover and after the production system is restored, the database on the production and backup systems must be synchronized. With synchronous remote journal, this can usually be achieved using the high availability application. With asynchronous remote journal, this may require resolving differences between the files and patching the backup database, or ignoring differences and using the backup database as is (see 5.4.1.1, “Recovery with Asynchronous Remote Journal” on page 88).

If a significant amount of activity occurs against the backup database before the production system returns, patching the backup database to preserve a relative record number synchronization between the files may not be feasible. In this

case, the backup database is hand-patched by inserting any records not found in the backup database at the next available record number. The databases contain the same data, but at different relative record locations. Because most high availability applications maintain synchronization between the databases using relative record numbers, the databases are not synchronized, at least from the high availability perspective.

The high availability application often has a built-in method for synchronizing the database after the patching process. To synchronize the database manually, save the database from the production system and restore it to the backup system. You can do this by using a communications link or a removable media. Figure 68 and Figure 69 illustrate the commands for the example application.

```
SAVRSTOBJ   OBJ(DB1) LIB(APPLIB) RMTLOCNAME(SRCSYS) MBROPT(*ALL)
            ALWOBJDIF(*ALL)
```

Figure 68. Synchronizing the Database Using Communications

```
On TGTSYS:
SAVOBJ OBJ(DB1) LIB(APPLIB) DEV(device-name)

On SRCSYS:
RSTOBJ OBJ(DB1) SAVLIB(APPLIB) DEV(device-name) MBROPT(*ALL)
        ALWOBJDIF(*ALL)
```

Figure 69. Synchronizing the Database Using Removable Media

5.6 Other Considerations for Switchover

A usable backup system usually requires more than just database files. The backup system must have all of the applications and objects required to continue critical business tasks and operations. Many of these objects can be created during the initial setup of the backup system. Users working in a dynamic environment may also need to maintain these objects in parallel on the production and backup systems. They may also have to account for changes in security for the objects.

5.6.1 User Access to the Backup System

Users require access to the backup system in a switchover situation. This may seem obvious, but access involves both connectivity and security issues. The users' devices must be able to connect to the system and the users must be able to sign on to the system. In some situations, these are non-trivial issues that require planning prior to an actual outage.

In a typical LAN environment with intelligent workstations, users may already have access to the backup system through an icon on their workstation desktop. The users switch to the backup system simply by clicking on the icon and signing on to that system.

In other cases, it may be desirable, or even necessary, to restrict access to the backup system until it is needed. This introduces complexity, since in an unplanned outage, users need quick access to the backup system. They do not want to have to wait for someone to come to each workstation to configure

access to the backup. A third-party switching solution that simplifies IP address takeover or SNA aliasing allows quick access to the backup in a controlled manner.

Devices directly connected to the production system, such as terminals attached to a workstation controller, may also need to be switched over for normal business activities to continue. Switching hardware, and an automatic switching solution that controls the hardware, may facilitate an easier transition. The switch can be performed for either planned or unplanned downtime with a minimal disruption to the end user. The high availability applications generally include this type of an automatic switching solution.

In both of the latter cases, the appropriate configuration objects (line, controller, and device objects) should exist on the backup system to allow the users to successfully access the system. These objects should exist prior to the switch and should be maintained in parallel with the production system, so that they have the same security and attributes.

Users who require access to the backup system following a switchover also require user profiles on the backup system. The user profiles on the backup system should reflect the current attributes of the user profiles on the production system, including the password. If access to the backup system must be restricted, the user profiles can be maintained in a disabled state and enabled when the switchover occurs.

If the backup system is used for another purpose under normal business conditions (for example, as an application development system), it may also be necessary to disable some or all of the regular users of the backup system. This alleviates performance impacts by the regular users on the critical business applications now running on the system, especially if the backup system is a smaller system than the production system.

5.6.2 Other Objects

The applications a business requires for its daily operations dictate what other objects are required on the backup system. Not all of the applications used during normal operations may be required on the backup system. In the event of an unplanned outage, the business may choose to run with a subset of those applications. This may allow the business to use a smaller system as the backup system or to lessen the impact of the additional users when the backup system is already used for other purposes.

The exact objects that comprise an application varies widely. The developers should be able to provide a comprehensive list of the objects used by in-house applications. Users may want to consult the application provider for third-party applications. Pay special attention to data storage objects other than the database files and also objects stored in multiple libraries or alternate file systems. Because this book focuses on remote journal, database files (*FILE) are assumed and are not included in the following list, which shows the object types that are commonly part of an application (the list is intended as a guideline only and is not comprehensive):

Authorization lists (*AUTL)	Authorization lists are used by some applications to simplify authorization to the application.
Data areas (*DTAARA)	Data areas are often used by older applications, especially applications written in RPG. They are often used as storage for runtime information or for interprocess communication.
Data queues (*DTAQ)	Data queues are often used by applications for interprocess communication. For some applications, the entries on the data queue are critical and should be replicated in realtime. For other applications, the existence of the data queue is sufficient, since the entries will be rebuilt when the application is restarted after a switchover.
Directories (*DIR)	Directories are the containers for stream files and symbolic links. The correct directories must exist for applications using these objects to function properly. Some applications create new directories at runtime.
Documents (*DOC)	Documents are used for data storage by OfficeVision/400 and by some older PC-based applications. They are commonly used as the intermediate storage medium in older imaging applications, prior to moving the image to optical storage.
Files (*FILE)	Other file types, such as logical files, display files, print files, and so on, may also need to exist for proper application operation. It may also be prudent to replicate source files for critical applications developed in-house to the backup.
Folders (*FLR)	Folders are the container for documents. The correct folders must exist for applications using documents to function properly. Some applications using documents create new folders at runtime.
Job description (*JOBD)	Applications that submit batch jobs may require application-specific job descriptions to exist for proper operation.
Libraries (*LIB)	The libraries that contain the other application objects obviously need to exist on the backup system. Some applications, especially change management products, may create new libraries during runtime.
Message queue (*MSGQ)	Applications that issue messages may require application-specific message queues to exist for proper operation.
Mode description (*MODD)	Applications that evoke communication jobs may require application-specific mode descriptions to exist for proper operation.

Programs (*PGM) Few, if any, applications do not involve programs that interface with the user and manipulate the database and other objects. All of the application's programs must exist to insure proper operation.

Service programs (*SRVPGM) Applications written in the OS/400 ILE languages may also use service programs, as well as programs. These must exist for the application to function properly.

Stream files (*STMF) Stream files are the binary data file object in the OS/400 Integrated File System (IFS). Stream files are used much the same as documents, especially for PC-based applications. They are also commonly used by applications that have been ported to the AS/400 system from other operating systems, such as the enterprise resource planning (ERP) applications SAP R/3 and Baan.

Symbolic links (*SYMLNK) Symbolic links are used within the Integrated File System to directly access a single object from multiple directory paths. They are often used by applications that are ported to the AS/400 system from other operating systems, such as the enterprise resource planning (ERP) applications SAP R/3 and Baan.

User Indices (*USRIDX) A user index can be used as an alternative to a database file for storing ordered lists with a simple access key, especially in performance sensitive applications.

User queues (*USRQ) User queues are similar to data queues in use and function and the same considerations apply. They are more likely to be used in applications written in the OS/400 ILE languages.

User spaces (*USRSPC) User spaces are commonly used to store configuration and run-time information. They are commonly used in applications written in the OS/400 ILE languages because they offer direct storage access using pointers.

For many of the objects in this list, the content, attributes, and security of the object affect how the application operates. The objects must be continuously synchronized between the production and backup systems. For some objects, replicating the object content in near realtime can be as important as replicating the database entries.

In cases where object synchronization is important, the user must consider whether they can adequately maintain the two systems in parallel manually. If not, the user must decided whether they can afford to develop an in-house application for automatic maintenance. Users must also consider the length of time that they can afford to be without their computing environment if the

production system is lost unexpectedly and manual updates must be applied to the backup before it can be used. In a highly dynamic environment or one where the cost of downtime rises rapidly, users may want to consider one of the third-party high availability applications available for the AS/400 system.

Chapter 6. AS/400 Switchover Solutions—Guidelines for ISVs

Mirroring data to one or more AS/400 systems can be made less complex and much easier to manage for the customer by following some guidelines. These guidelines are discussed in the following sections.

6.1 Determining What must be Journaled

In general, customers select objects to be mirrored by a library. In large environments, it can be difficult to determine which libraries should be mirrored. (In many cases, because of overhead it is not practical or required to mirror all libraries.) This process can be made less dramatic by ensuring that relational objects such as physical and logical files are contained in the same library so the customer does not have to search for them. Typically, a high availability solution includes a seamless configuration of mirrored libraries and objects.

Many applications use temporary or work files that are not required to be mirrored. If they are mirrored, the applications use additional CPU cycles. Using a naming convention for these work files that allows generic selection or omission allows the customer to easily identify and exclude them from the mirroring and journaling process. This reduces overhead. Another even better solution is to place work files in a separate library that will not be journaled or mirrored at all. The files should be created by the process if they do not exist in the work library.

6.2 Creating and Deleting Data Objects

Deleting and re-creating data objects often presents timing issues for high availability solutions. When architecting software applications to be implemented in a switchover environment, it is best to ensure that data objects, such as files, data areas, and so on, remain on the system. Object creation operations are not journaled to the user journal and must be retrieved by other asynchronous processes. A preferable solution is to use the CLRPFM command instead of deleting and re-creating an existing file.

6.3 ENDJRNPF and STRJRNPF Commands

Since most high availability solutions require the journaling of database files, software vendors should avoid ending and starting journaling outside of the high availability product. If done at an inappropriate time, ending journaling for a file causes transactions to be lost to the high availability solution. This may results in synchronization issues.

6.4 Processing under Commitment Control

Using commitment control within applications can assist in the recovery process in the event of a system failure where the target system must be relied on to continue business operations. Commitment control is used to mark the boundaries of a *block* of transactions that are related. An order and all line items in the order are related because without all of the line items present, the order is not complete.

If the order entry application does not use commitment control and the source system fails, some of the line item records may be lost. The records that are actually written to the line item file (disk) are, therefore, journaled and sent to the target system. After a delay, business resumes on the target system. Typically, the clerk reviews the last order to insure it was present before continuing with the next order. Because the order is present, the clerk may not realize that the order is missing items until the customer receives the order and brings it to their attention.

In the same situation using commitment control, all of the transactions for the order, including the order itself, are rolled back out of the database before any of the users are allowed on the system (typically done by the high availability software). This happens because no end to the commitment boundary is present. When the clerk verifies the last order entered, it is easier to identify the incomplete order since it is not present. Hopefully, this prompts the clerk to re-contact the customer.

In the following example, when the application is under commitment control, the first journal entry generated (journal code C, entry type BC) identifies the beginning of the commitment control transaction. Record PT1 is written to the Order Master File followed by records PT1, PT2, PT3, and PT4, which are written to the Order Line Item File. The application issues the commit instruction which closes the transaction. This is denoted by the journal code C, entry type CM journal entry.

Note

Files must be journaled to use commitment control.

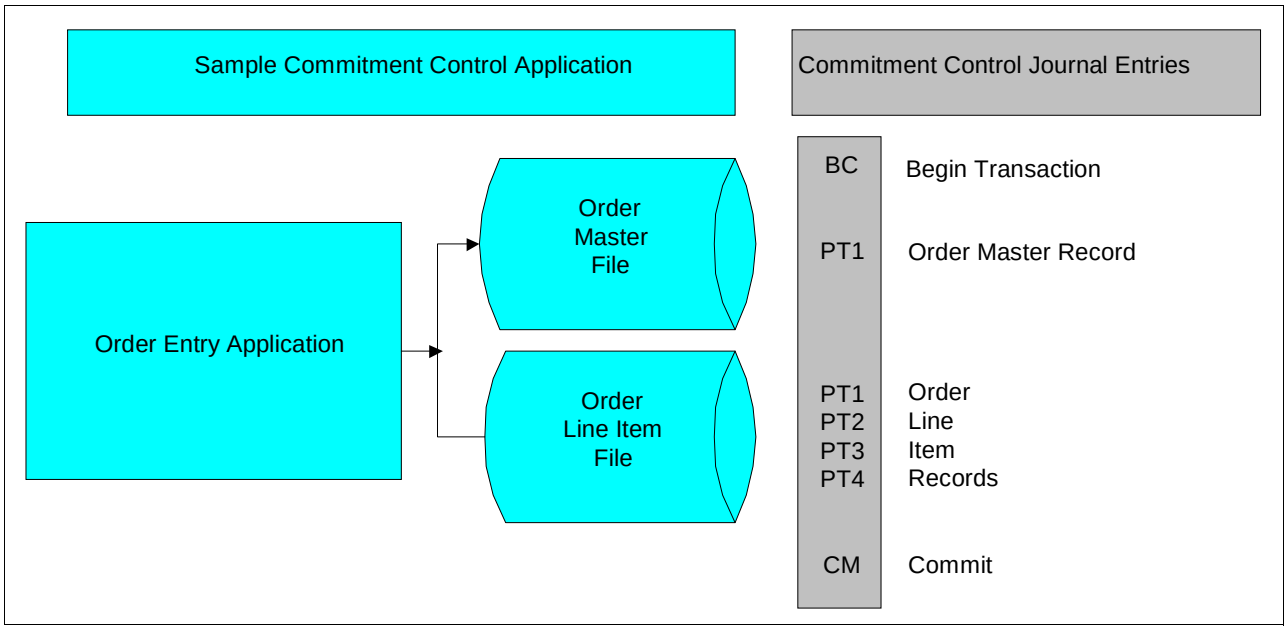


Figure 70. Sample Application Using Commitment Control

In the sample application illustrated in Figure 70, if the source system fails after PT2 is written to the Order Line Item File, the CM journal entry would never reach

the journal on the target system because it did not occur on the source system. On the target, when the switchover begins, the application can detect the uncommitted transactions and can roll back to the beginning of the transaction (BC journal entry). This is all completed before the users are allowed on the target system to resume work.

6.5 High Availability Solutions

While designing you application please keep in mind that there are many other pieces to the High Availability solution in addition to replicating database files, which include but are not limited to:

- Transaction-level object selection and management
- Inquiry and monitoring functions of the replication system
- User profile and password replication
- Non-transaction based object replication
- Switchover processes from the source system to the target system and vice versa
- Authority mirroring

Consider the following good practices when implementing an HA solution:

- Minimize move and rename operations for the application objects.
- Try to avoid long running commitment control transactions.
- Consider the implications of using system managed journals.

Contact IBM or your local vendor for further details.

Appendix A. Benchmark Scenario

This appendix lists the steps and commands we used to run the remote journal performance benchmark over different communications gear.

The initial setup for the benchmark is shown in Figure 71.

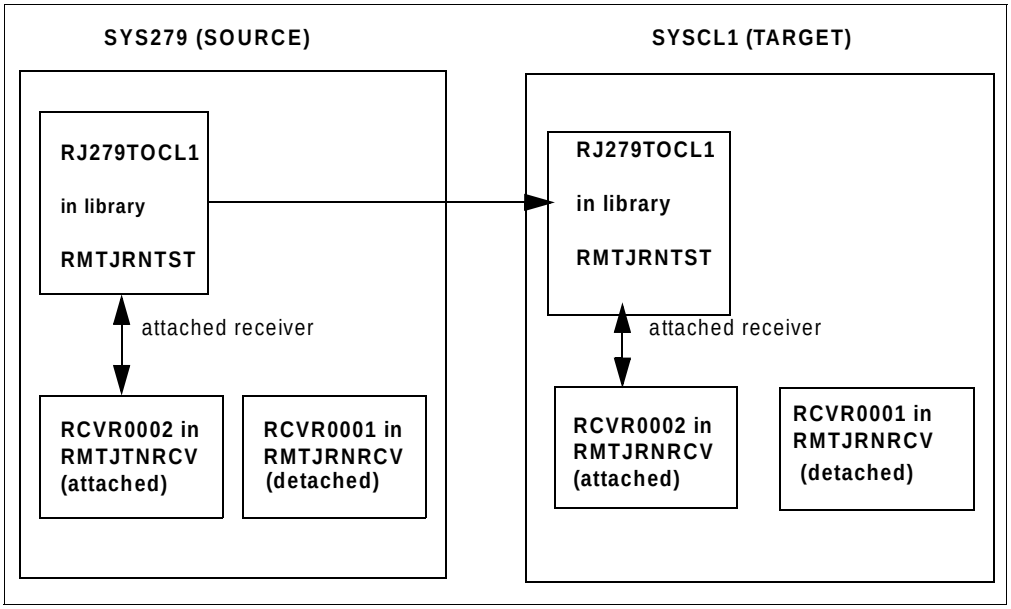


Figure 71. Remote Journal Setup for the Performance Benchmark

The following steps describe the test scenario we used for different communications links and delivery modes:

1. Inactivate the remote journal on the source system (SYS279). We use separate RDB directory entries for different communications links. The RDB entry SYSCL1ATM denotes a connection over 155Mb ATM from SYS279 to SYSCL1.

```
CHGRMTJRN RDB(SYSCL1ATM) +
          SRCJRN(RMTJRNTST/RJ279TOCL1) +
          JRNSTATE(*INACTIVE) +
          INACTOPT(*IMMED) +
```

Figure 72. Inactivate Remote Journal on the Source System

2. Remove the remote journal association on the source system.

```
RMVRMTJRN RDB(SYSCL1ATM) +
          SRCJRN(RMTJRNTST/RJ279TOCL1)
```

Figure 73. Delete Remote Journal Association on the Source System

3. Delete the remote journal and receivers on the target system (SYSCL1). This step was required since we wanted to make sure that each test run started in exactly the same environment. We learned that leaving the detached receivers in the target system's user ASP can influence the speed of writing new entries to the attached receivers. There was a difference if we created a receiver in an empty user ASP, or the ASP was half full. The explanation for this phenomenon is that the average search for a disk arm slightly changes when the disk is filling up.

```
DLTJRN JRN(RMTJRNTST/RJ279TOCL1)
DLTJRNRCV JRNRCV(RMTJRNRCV/RCVR0001)
```

Figure 74. Remove Remote Journal and Receiver on the Target System

4. Change the local journal, and generate a new receiver on the source system. In this step, we created a new receiver and reset the journal sequence number in the newly attached receiver. Please note the receiver name was generated by the system.

```
CHGJRN JRN(RMTJRNTST/RJ279TOCL1) +
       JRNRCV(*GEN) +
       SEQOPT(*RESET)
```

Figure 75. Change Journal on the Source System

5. Add a remote journal on the source system. We created a separate message queue for the test environment to isolate the remote journal messages from other system messages.

```
ADDRMTJRN RDB(SYSCL1ATM) +
          SRCJRN(RMTJRNTST/RJ279TOCL1) +
          MSGQ(RMTJRNTST/RMTJRNQ)
```

Figure 76. Add Remote Journal on the Source System

6. Activate the remote journal. Depending on the test scenario, we activated the remote journal either in synchronous or asynchronous mode.

```
CHGRMTJRN RDB(SYSCL1ATM) +
          SRCJRN(RMTJRNTST/RJ279TOCL1) +
          JRNSTATE(*ACTIVE) +
          DELIVERY(*SYNC)
          STRJRNRCV(*ATTACHED)
```

Figure 77. Activate Remote Journal on the Source System

7. Run the benchmark. All of the tests were run with commitment control.

Appendix B. Special Notices

This publication is intended to help system administrators, high availabilty specialists, and Independent Software Vendors (ISVs) to efficiently use remote journal function for high availabilty solutions on the AS/400 platform. The information in this publication is not intended as the specification of any programming interfaces that are provided by the Operationg System/400. See the PUBLICATIONS section of the IBM Programming Announcement for OS/400 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was obtained in a controlled environment based on the use of specific data and is presented only to illustrate techniques and procedures to assist IBM personnel to better understand IBM products. The results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data in their

specific environment. No performance data may be abstracted or reproduced and given to non-IBM personnel without prior written approval by Business Practices.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain 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.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

APPN	AS/400
DB2	Distributed Relational Database
DRDA	Architecture
IBM®	OfficeVision/400
OS/400	400

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 107.

- *DB2/400 Advanced Database Functions,* SG24-4249
- *The System Administrator’s Companion to AS/400 Availability and Recovery*, SG24-2161

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *Backup and Recovery*, SC41-5304
- *OptiConnect for OS/400*, SC41-5414
- *SNA Distribution Services*, SC41-5410
- *TCP/IP Configuration and Reference*, SC41-5420
- *OS/400 Distributed Database Programming*, SC41-5702
- *OS/400 LAN, Frame Relay and ATM Support*, SC41-5404
- *OS/400 Communications Configurations*, SC41-5401

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com/`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/`

- **PUBORDER** – to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** – send orders to:

	IBMMAIL	**Internet**
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** – send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** – send orders to:

United States (toll free)	1-800-445-9269
Canada	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 408 256 5422 (Outside USA)** – ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name | Last name

Company

Address

City | Postal code | Country

Telephone number | Telefax number | VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date | Card issued to | Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

AR	Application Requester
ATM	Asynchronous Transfer Mode
AS	Application Server
CPU	Central Processing Unit
DASD	Direct Access Storage Device
DBMS	Database Management System
DDI	Distributed Data Interface
DDM	Distributed Data Management
DDS	Data Definition Specification
DRDA	Distributed Relational Database Architecture
HA	High Availability
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
LAN	Local Area Network
MI	Machine Interface
OS/400	Operating System/400
PTF	Program Temporary Fix
RDB	Relational Database
SLIC	System License Internal Code
SQL	Structured Query Language
SNA	System Network Architecture
TCP/IP	Transmission Control Protocol/Internet Protocol

Index

Symbols

A

B

C

D

E

G

H

V

W

ITSO Redbook Evaluation

AS/400 Remote Journal Function for High Availability and Data Replication
SG24-5189-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ **Customer** _ **Business Partner** _ **Solution Developer** _ **IBM employee**
_ **None of the above**

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

_____

SG24-5189-00

Printed in the U.S.A.

AS/400 Remote Journal Function for High Availability and Data Replication

SG24-5189-00