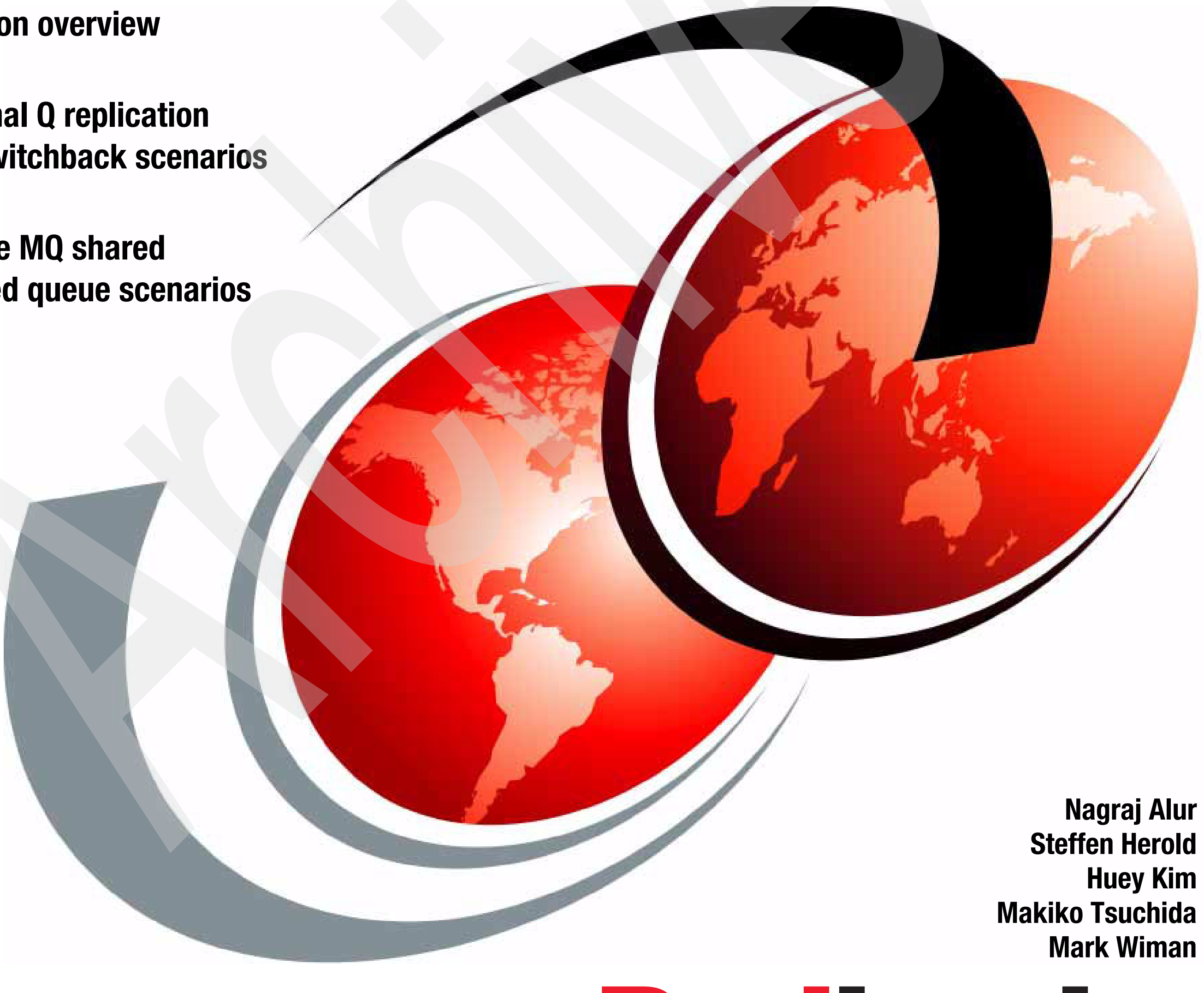
WebSphere Replication Server for z/OS Using Q Replication

High Availability Scenarios for the z/OS Platform

WebSphere Replication Server for z/OS Q replication overview

Bidirectional Q replication failover/switchback scenarios

WebSphere MQ shared disk/shared queue scenarios



Nagraj Alur
Steffen Herold
Huey Kim
Makiko Tsuchida
Mark Wiman

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere Replication Server for z/OS Using Q Replication: High Availability Scenarios for the z/OS Platform

June 2006

SG24-7215-00

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

First Edition (June 2006)

This edition applies to Version 9 of WebSphere Replication Server for z/OS (product number 5655-R55).

Contents

Figures

Tables

Examples

Notices

Trademarks

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

@server	DB2	OS/390
@server	Informix	Parallel Sysplex
Redbooks (logo)	IBM	Redbooks
z/OS	MQSeries	WebSphere
AIX	MVS	

The following terms are trademarks of other companies:

Windows™, and the Windows logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Linux™ is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbook documents the procedures for implementing WebSphere® Information Integrator's Q replication technologies in support of high availability scenarios involving bidirectional and unidirectional replication solutions.

This book is aimed at an audience of IT architects and database administrators (DBAs) responsible for developing high availability solutions on the z/OS® platform.

This book documents a step-by-step approach to implementing high availability scenarios involving bidirectional Q replication and WebSphere MQ shared disk and WebSphere MQ shared queue technologies.

Important: This book uses the next release of Q replication in all its scenarios. While we do not have these scenarios with the current V8.3 release, we fully expect the scenarios and approaches documented here to apply.

This book is organized as follows:

- Chapter 1, "High availability concepts" on page 1, provides an introduction to high availability concepts and describes some of the terminology used in this book.
- Chapter 2, "WebSphere Replication Server Q replication overview" on page 7, provides an overview of Q replication, its architecture, processing flow, and key considerations in choosing a particular Q replication topology to address a business requirement.
- Chapter 3, "Failover and switchback scenarios" on page 27, describes a step-by-step approach to performing failover and switchback in a bidirectional replication scenario involving two z/OS servers.
- Chapter 4, "WebSphere MQ shared queues and unidirectional replication" on page 103, describes a step-by-step approach to implementing a WebSphere MQ shared queues high availability environment on the source system of a unidirectional Q replication environment.
- Chapter 5, "WebSphere MQ shared disk and unidirectional replication" on page 165, describes a step-by-step approach to implementing a WebSphere MQ shared disk high availability environment on the source system of a unidirectional Q replication environment.
- Appendix A, "Summary of code and scripts used in the scenarios" on page 189, summarizes the code and scripts used in the scenarios that can be downloaded from the IBM Redbooks Web site.
- Appendix B, "Exception processing in a bidirectional Q replication environment" on page 191, describes exception processing in Q replication.
- Appendix C, "WebSphere shared queues and shared disk overview" on page 251, provides an overview of WebSphere shared queues and WebSphere shared disk technologies, and summarizes the considerations in choosing one or the other in a high availability environment.

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, San Jose Center.

Nagraj Alur is a Project Leader with the IBM International Technical Support Organization, San Jose Center. He holds a Masters Degree in Computer Science from the Indian Institute of Technology (IIT), Mumbai, India. He has more than 30 years of experience in DBMSs and has been a programmer, systems analyst, project leader, independent consultant, and researcher. His areas of expertise include DBMSs, data warehousing, distributed systems management, database performance, information integration, as well as client/server and Internet computing. He has written extensively on these subjects and has taught classes and presented at conferences all around the world. Before joining the ITSO in November 2001, he was on a two-year assignment from the Software Group to the IBM Almaden Research Center, where he worked on Data Links solutions and an eSourcing prototype.

Steffen Herold is an IBM Certified DBA for DB2® UDB, working as a Support Analyst in the DB2 UDB EMEA Advanced Support Group, in Munich, Germany. He has more than 10 years of experience in supporting customers across Europe on UNIX® and Linux® platforms with high-end OLTP and Data Warehouse databases. His area of expertise include high availability, performance tuning, backup/restore, crash analysis, and debugging for IBM DB2 UDB and Informix® database servers. Steffen holds a degree in Computer Science from Technical University of Chemnitz, Germany.

Huey Kim is a Database Administrator for IBM Integrated Technology Delivery, Server Operations in New York. He has seven years of experience in the IT industry, and he holds a Bachelor of Science degree in geology from Binghamton University in Binghamton, New York. His areas of experience include database performance, data replication, and problem determination in DB2 UDB for z/OS and for Linux, UNIX, and Windows®.

Makiko Tsuchida is an IT Specialist with IBM Systems Engineering Co.,Ltd. in Japan. She has been in technical support for information management products for four years. She has taught workshops on SQL Replication and WebSphere Information Integrator. She was involved in building Japan's first Q Replication system at the Kyoto University Hospital in 2005. She also participated in Q Replication new function unit testing at the IBM Silicon Valley Laboratory, San Jose in March 2005.

Mark Wiman is a certified IT Specialist with IBM Business Consulting Services (Application Services). He holds a Master's Degree in Computer Science from the Georgia Institute of Technology. He has more than 28 years of experience in the IT industry where he has served as a systems programmer, project manager, application developer, application architect, and database administrator. He currently serves as a Worldwide DBA for the internal IBM implementation of Siebel and is currently focused on implementing a single worldwide instance of Siebel Analytics.

We gratefully acknowledge the content we incorporated from other IBM Redbooks such as *WebSphere Application Server V6: High Availability Solutions*, SG24-6688, for high availability concepts; and *WebSphere MQ in a z/OS Parallel Sysplex Environment*, SG24-6864, and *WebSphere MQ V6 Fundamentals*, SG24-7128, for shared queues information.

Thanks to the following people for their contributions to this project:

Jaime Anaya
Dell Burner
Beth Hamel

Tom Jacopi
Donna J Kelsey
Madhu Kochar
Somil Kulkarni
Kevin Lau
Jayanti Mahapatra
San Phoenix
Patricia Shimer
Asim Singh
IBM Silicon Valley Laboratory, San Jose

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 026
  5600 Cottle Road
  San Jose, California 95193

1

High availability concepts

In this chapter we introduce high availability concepts and describe some of the terminology used in this book.

1.1 Introduction

This book describes high availability (HA) implementations on the z/OS platform using WebSphere Information Integrator Q Replication and WebSphere MQ shared disk and shared queue technologies. It is therefore desirable to define the term *availability* and the various levels of availability that exist.

System[1] availability is a measure of the time that the system is functioning normally, as well as a measure of the time required to recover the system after it fails. In other words, it is the downtime that defines system availability. This downtime includes both planned and unplanned downtime.

Let "A" be an index of system availability expressed as a percentage, MTBF the mean time between failures, and MTTR the maximum time to recover the system from failures. Thus, we have:

```
A = MTBF/(MTBF + MTTR)
```

This formula shows that:

- As MTBF gets larger, "A" increases and MTTR has less of an impact on "A".
- As MTTR approaches zero, "A" approaches 100%.

This means that if we recover from failures very quickly, we have a highly available system. The time to restore a system includes:

- Fault detection time
- Service recovery time

Any high availability implementation must therefore seek to minimize both these items. Technologies such as clustering software use fault detection mechanisms and automatically fail over the services to a healthy host to minimize fault detection time and service recovery time.

Important: Implicit in any discussion of high availability is the issue of *data loss*. This refers to the amount of electronically stored business process information in the failing system that may be temporarily or permanently lost during the system recovery. The amount of data loss may be a very important consideration when evaluating alternative high availability implementations.

It should be noted that the formula provided is a rather simplistic measure of availability for the following reasons:

- MTBF is just a statistical measure. For example, if a CPU has an MTBF of 500,000 hours, it does not necessarily mean that this CPU will fail after 57 years of use. In practice, this CPU can fail either before or after this interval.
- A system typically consists of a very large number of components, with each component potentially having a different MTBF and MTTR.

Therefore, system availability is very hard to predict and is usually determined by the weakest component in a system.

Usually, redundant hardware and clustering software are used to achieve high availability. The goal should be to minimize the MTTR through various high availability techniques. If

[1] A system typically includes hardware, software, data, and processes.

MTTR approaches zero, then system availability approaches 100%, no matter what the MTBF is.

In the following section we describe the various levels of availability and define the terms *high availability*, *continuous operation*, and *continuous availability*.

1.2 Levels of availability

There is a direct correlation between the level of availability and the cost of providing it, as shown in Figure 1-1. The higher the investment, the greater the availability (less downtime). Individual organizations need to weigh the cost/benefits of providing high levels of availability for their particular systems.

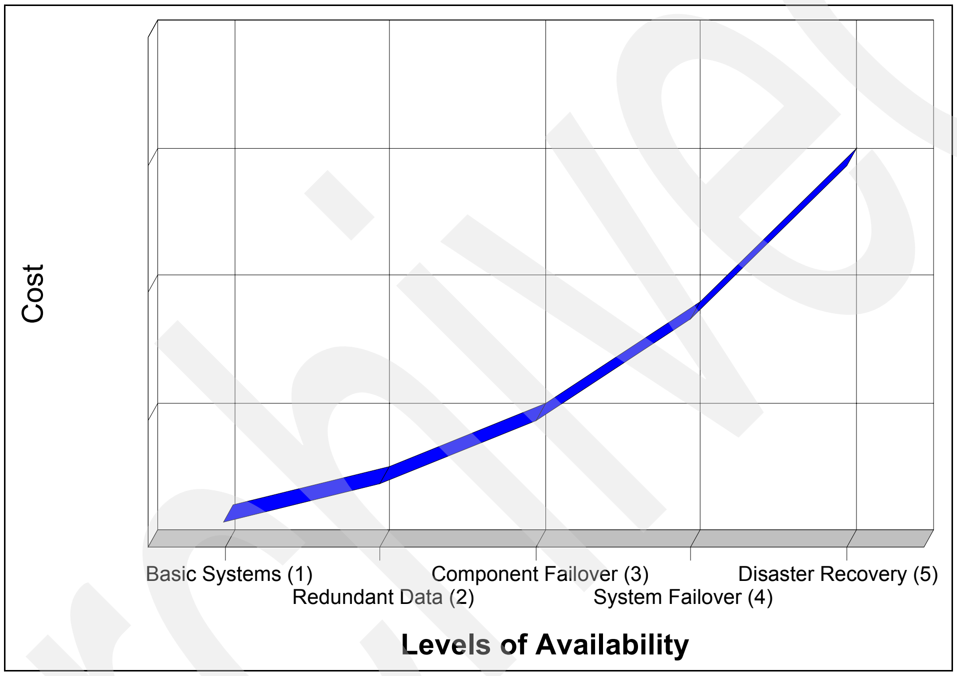


Figure 1-1 Levels of availability and costs

Figure 1-1 shows five levels of availability as follows:

1. Basic systems

   Basic systems do not employ any special measures to protect data and services, although backups are taken regularly. When an outage occurs, support personnel usually restore the system from the most recent backup.

2. Redundant data

   Disk redundancy or disk mirroring is used to protect the data against the loss of a disk. Full disk mirroring provides more data protection than RAID-5.

3. Component failover

   Most systems consist of many components such as firewalls, load balancers, Web servers, application servers, database servers, and security servers. An outage in any of these components can result in service interruption. Multiple threads or multiple instances can be employed for availability purposes. For example, if you do not make the firewall component highly available, it might cause the whole system to go down. Worse still is the potential exposure of the system to hackers, even though the servers are highly available.

Highly available data management is critical for a highly available transactional system. Therefore, it is very important to balance the availability of all components in a system. Do not overspend on any particular component or underspend on other components. For the system shown in Figure 1-2, the system availability seen by the client would be 85%.

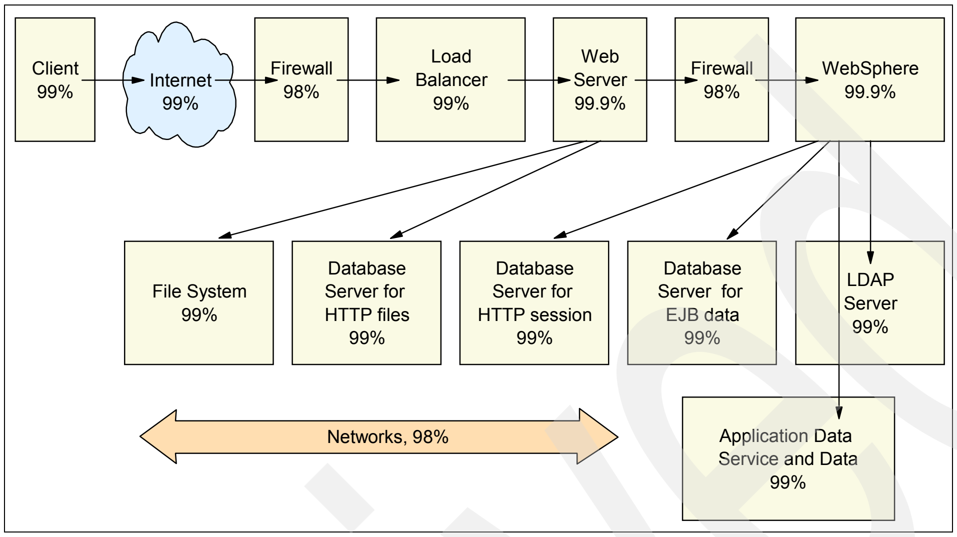

Client 99%
Internet 99%
Firewall 98%
Load Balancer 99%
Web Server 99.9%
Firewall 98%
WebSphere 99.9%
File System 99%
Database Server for HTTP files 99%
Database Server for HTTP session 99%
Database Server for EJB data 99%
LDAP Server 99%
Networks, 98%
Application Data Service and Data 99%

Figure 1-2 Availability chains

4. System failover

   A standby or backup system is used to take over for the primary system if the primary system fails. In principle, almost any kind of service can become highly available by employing system failover techniques.

   Typically in system failover, clustering software monitors the health of the network, hardware, and software processes; detects and communicates any fault; and automatically fails over the service and associated resources to a healthy host. This allows the service to continue more or less uninterrupted before repairing the failed system.

   The two systems involved may be configured as Active/Active[2] mutual takeover or Active/Passive[3] takeover.

   Note: System failover may also be used for planned software and hardware maintenance and upgrades.

5. Disaster recovery

   This applies to maintaining systems at different sites to guard against natural or human-made catastrophic events that may cause the primary site to be completely destroyed. In such cases, the backup site becomes the new primary operational site within a reasonable time. There is generally some data loss, and no reverting back to the primary site, which is not recoverable.

Table 1-1 provides a perspective on availability in terms of hours/minutes/seconds of downtime. For example:

- 99% availability corresponds to a downtime of 14 minutes per day on average.

[2] Both the primary and the secondary are actively servicing requests.
[3] Only the primary is servicing requests, while the secondary waits passively to take over the servicing of requests in the event the primary fails.

- 99.9% availability corresponds to a downtime of 1.4 minutes per day on average.

Many people refer to 99%, 99.9%, 99.99%, and 99.999% as “two nines,” “three nines,” “four nines,” and “five nines.”

“Five nines” is generally thought of as the best achievable system with reasonable costs, and many vendors offer such solutions. “Five nines” availability allows a downtime of 864 milliseconds per day, 6 seconds per week, and 5.3 minutes per year, as shown in Table 1-1.

Table 1-1 Availability metric - nine rule

9s	**Percentage of uptime**	**Downtime per year**	**Downtime per week**	**Downtime per day**
	90%	36.5 days	16.9 hours	2.4 hours
	95%	18.3 days	8.4 hours	1.2 hours
	98%	7.3 days	3.4 hours	28.8 minutes
Two 9s	99%	3.7 days	1.7 hours	14.4 minutes
	99.5%	1.8 days	50.4 minutes	7.2 minutes
	99.8%	17.5 hours	20.2 minutes	2.9 minutes
Three 9s	99.9%	8.8 hours	10.1 minutes	1.4 minutes
Four 9s	99.99%	52.5 minutes	1 minute	8.6 seconds
Five 9s	99.999%	5.3 minutes	6 seconds	864 milliseconds
Six 9s	99.9999%	31.5 seconds	604.8 milliseconds	86.4 milliseconds
Seven 9s	99.99999%	3.2 seconds	60.5 milliseconds	8.6 milliseconds
Eight 9s	99.999999%	315.4 milliseconds	6 milliseconds	0.9 milliseconds

The industry standard definition of the terms “high availability,” “continuous operation,” and “continuous availability” are as follows:

- High availability (HA)

  This refers to the characteristic of a system that delivers an acceptable or agreed-upon level of service during scheduled periods. The availability level can be stated in a service level agreement between the end users’ representative and the service provider; this is the “agreed level.” The fact that the service is available only during scheduled periods implies that there is time available to apply changes and maintenance outside these periods. When discussing high availability systems we often end up discussing such items as redundant hardware and software subsystems, backup channel paths, RAID DASD, and techniques to reduce the number of single points of failure.

- Continuous operation (CO)

  This is the characteristic of a system that operates 24 hours a day, 365 days a year with no scheduled outages. This does not imply that this system is highly available. An application could run 24 hours a day, 7 days a week and be available only 95% of the time if there were a high number of unscheduled outages. When discussing continuous operation for systems, we often end up discussing such items as techniques for changing the system or application without shutting down the service, or fast switch-over to a backup or stand-in service.

- Continuous availability (CA)

  This is high availability, extended to 24 hours a day and 365 days a year. CA combines the characteristics of continuous operations and high availability, to mask or eliminate all planned (scheduled) and unplanned (unscheduled) outages from the end user.

Important: Availability must be measured at the level of the end user, who is concerned only with getting his/her work done. In order for that to happen, the applications must be operational and available to the end user, including the hardware, software, network resources, and data required to run the applications.

2

WebSphere Replication Server Q replication overview

In this chapter we provide an overview of Q replication and its architecture and processing flow, discuss key considerations in choosing a particular Q replication topology to address a business requirement, and provide best practices implementation considerations.

The topics covered are:

- Q replication overview
- Q replication processing flow
- Choosing a particular Q replication topology
- Latency considerations

2.1 Q replication overview

Q replication is a high-volume, low-latency replication solution that uses WebSphere MQ message queues to transmit transactions between source and target databases or subsystems. Figure 2-1 shows a simple configuration of Q replication.

Important: WebSphere Replication Server for z/OS was formerly called WebSphere Information Integrator Replication for z/OS.

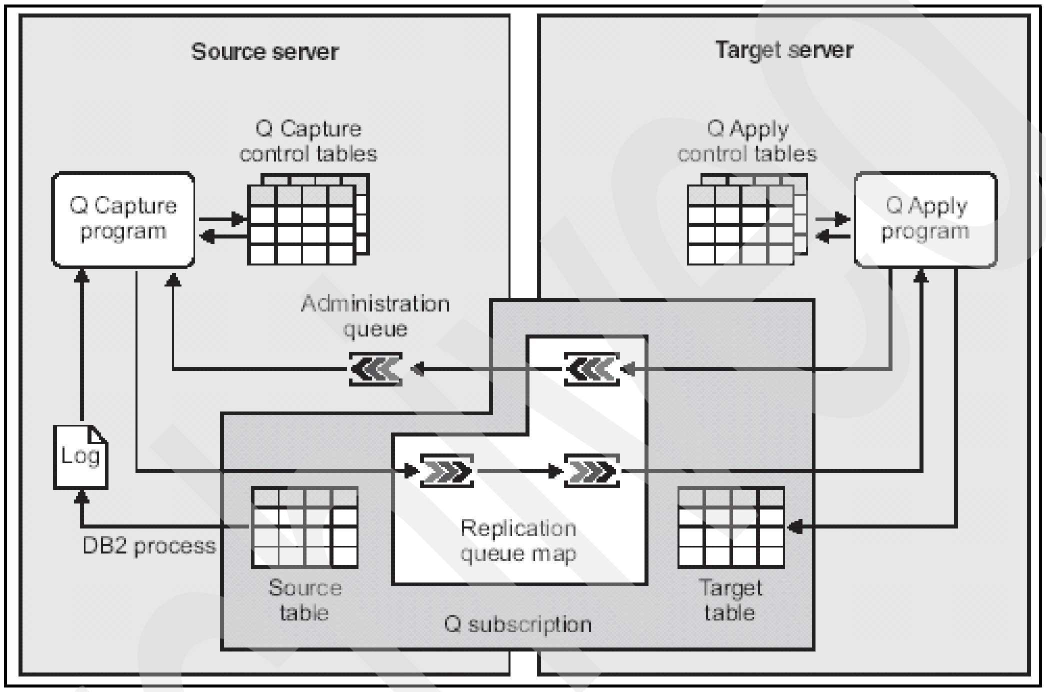


Figure 2-1 A simple Q replication configuration

The Q Capture program reads the DB2 recovery log for changes to a source table that is to be replicated. The program then sends transactions as messages over queues, where they are read and applied to target tables by the Q Apply program.

Note: Multiple tables are usually replicated through a single queue.

This type of replication offers several advantages, as follows:

- Minimum latency

  Changes are sent as soon as they are committed at the source and read from the log.

- High-volume throughput

  The Q Capture program can keep up with rapid changes at the source, and the multi-threaded Q Apply program can keep up with the speed of the communication channel.

- Minimum network traffic

  Messages are sent using a compact format, and data-sending options allow one to transmit the minimum amount of data.

- Asynchronous

  The use of message queues allows the Q Apply program to receive transactions without having to connect to the source database or subsystem. Both the Q Capture and Q Apply programs operate independently of each other—neither one requires the other to be operating. Of course, changes cannot be replicated unless they are captured by the Q Capture program and written to the message queues and retrieved and applied to the target by the Q Apply program. If the Q Apply program is stopped, messages remain on queues to be processed whenever the program is ready. If the Q Capture program is stopped, the Q Apply program continues to process messages on the queues. The messages are persistent and survive a system or device failure.

Q replication allows many different configurations. One can replicate between remote servers or within a single server. One can replicate changes in a single direction or in multiple directions. Replicating in multiple directions can be bidirectional (useful for managing standby or backup systems) or peer-to-peer (useful for synchronizing data on production systems).

The following objects are required for Q replication:

- Q replication control tables, which store information about Q subscriptions and XML publications, message queues, operational parameters, and user preferences.
- Replication queue maps, which identify the WebSphere MQ queues to be used for sending and receiving data
- Q subscriptions, which identify the source and target tables as well as options such as the rows and columns to be replicated or published, and the options for loading target tables
- WebSphere MQ queue managers, sender and receiver channels, transmit queues, and model/local/remote queues

For a complete description of these Q replication objects, refer to the DB2 Information Center:

http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp

Q replication supports three types of replication, as follows:

- Unidirectional replication
- Bidirectional replication
- Peer-to-peer replication

The following subsections provide a quick overview of these three types of Q replication.

2.1.1 Unidirectional replication

Unidirectional replication is a configuration that has the following characteristics:

- Changes that occur at a source table are replicated over WebSphere MQ queues to a target table, consistent change data (CCD[1]) table, or are passed as input parameters to a stored procedure to manipulate the data.
- Changes that occur at the target table are not replicated back to the source table.
- The target table typically is read-only, or is updated only by the Q Apply program. Conflict detection and resolution is supported.
- The source and target tables may be on the same server or different servers.
- Filtering is supported.

[1] The consistent-change data (CCD) table is a type of replication target table that is used for storing history, for auditing data, or for staging data. A CCD table can also be a replication source. There are different types of CCD tables such as complete CCD table, condensed CCD table, external CCD table, internal CCD table, noncomplete CCD table, and noncondensed CCD table.

2.1.2 Bidirectional replication

Bidirectional replication is a configuration that has the following characteristics:

- Replication occurs between tables on two servers. Changes that are made to one copy of a table are replicated to a second copy of that table, and changes that are made to the second copy are replicated back to the first copy.
- Updates on either of the servers are replicated to the other server.
- Filtering is not supported.
- Applications on any of the servers can update the same rows in those tables at the same time. However, this type of replication is chosen when there is generally little or no potential for the same data in the replicated tables to be updated simultaneously by both servers. Either the same row is considered to be updated by one server at a time, or one server updates only certain columns of data, and the other server updates the other columns.
- However, one can choose which copy of the table wins in the event a conflict happens to occur.

2.1.3 Peer-to-peer replication

Peer-to-peer replication (also known as multi-master replication) is a configuration that has the following characteristics:

- Replication occurs between tables on two or more servers.
- Updates on any one server are replicated to all other associated servers that are involved in the peer-to-peer configuration.
- Filtering is not supported.
- Applications on any of the servers are assumed to update the same rows and columns in those tables at the same time.
- All servers are equal peers with equal ownership of the data—no server is the “master” or source owner of the data. In a conflict situation, the change with the latest timestamp is the victor.

2.2 Q replication processing flow

The IBM Redbook *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487, describes a step-by-step procedure for setting up a bidirectional and peer-to-peer Q replication environment on the z/OS and AIX® platforms.

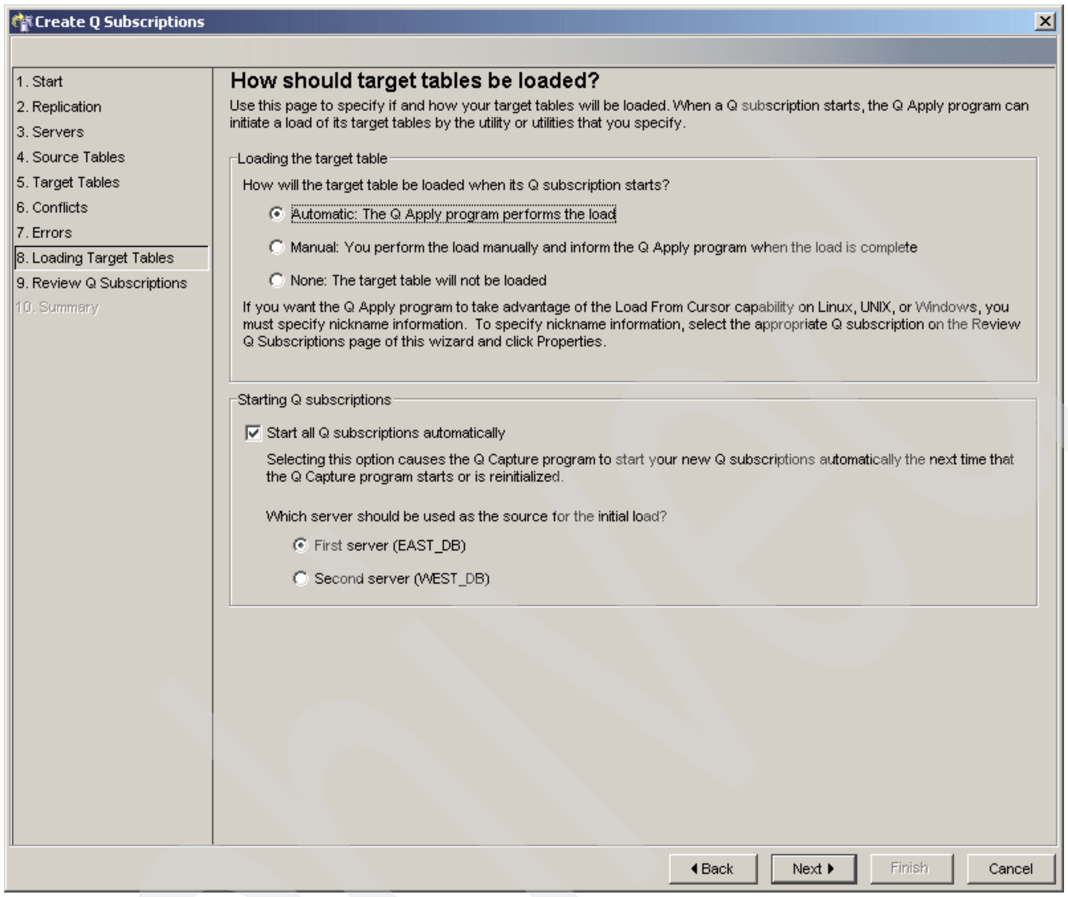
In this section, however, we focus on the processing flow that occurs when the Q subscription is first activated in a two-server configuration. This flow falls into two broad phases, as follows:

- Initial synchronization of the data at the source and target
- Ongoing replication after the initial synchronization

2.2.1 Initial synchronization of the data at the source and target

The activation of a subscription triggers the initial load and the ongoing replication process between the source and the target.

During the configuration of Q subscriptions using the Replication Center, one can specify how the target table is to be loaded (automatic, manual, or none), as shown in Figure 2-2.



How should target tables be loaded?

Use this page to specify if and how your target tables will be loaded. When a Q subscription starts, the Q Apply program can initiate a load of its target tables by the utility or utilities that you specify.

Loading the target table

How will the target table be loaded when its Q subscription starts?

[✓ Automatic: The Q Apply program performs the load]

Manual: You perform the load manually and inform the Q Apply program when the load is complete

None: The target table will not be loaded

If you want the Q Apply program to take advantage of the Load From Cursor capability on Linux, UNIX, or Windows, you must specify nickname information. To specify nickname information, select the appropriate Q subscription on the Review Q Subscriptions page of this wizard and click Properties.

Starting Q subscriptions

[✓] Start all Q subscriptions automatically

Selecting this option causes the Q Capture program to start your new Q subscriptions automatically the next time that the Q Capture program starts or is reinitialized.

Which server should be used as the source for the initial load?

[✓ First server (EAST_DB)]

Second server (WEST_DB)

Figure 2-2 Specifying how target table is to be loaded for given Q subscription

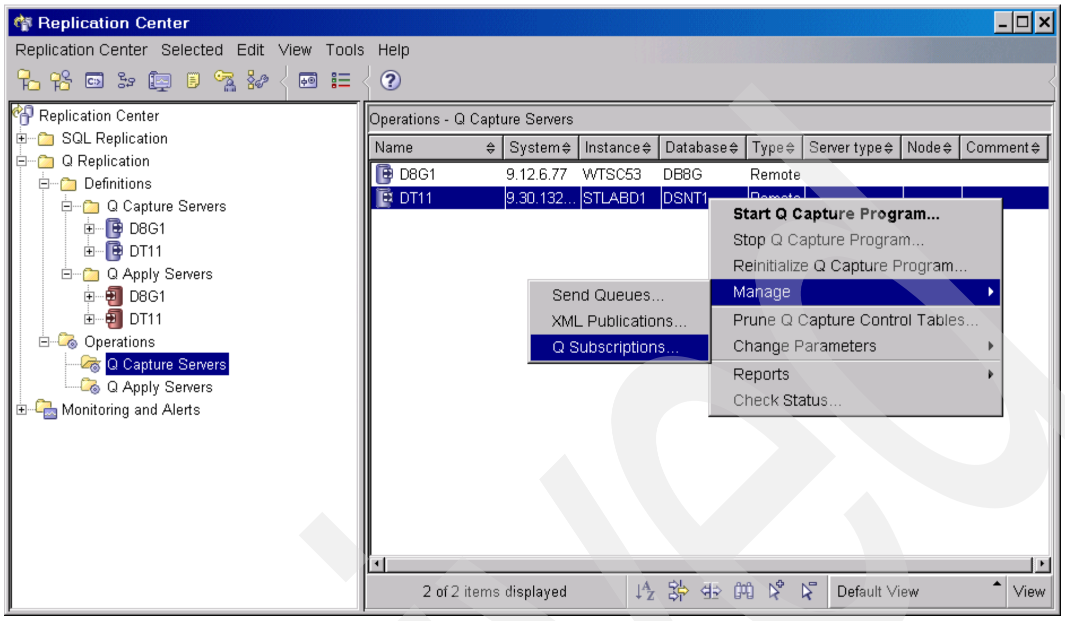
The Replication Center and ASNCLP commands generate SQL that records the new state "N" in the STATE column of the IBMQREP_SUBS table. When Q Capture sees the "N" state during startup or after being re-initialized, it activates them as if CAPSTART signals were received on them. The insert into the IBMQREP_SIGNAL table gets logged, and the Q Capture program initiates the processing for this Q subscription when it encounters this log record as it scans the log for changes.

Note: A new subscription may be activated when Q Capture is running, or a deactivated Q subscription may be reactivated by a user via an insert into the IBMQREP_SIGNAL table, as shown in Example 2-1. It may also be activated via the Manage Q Subscriptions window in the Replication Center, as shown in Figure 2-3, which causes the insert to be generated into the IBMQREP_SIGNAL table.

Example 2-1 Insert a CAPSTART signal in the IBMQREP_SIGNAL table

```
insert into capture_schema.IBMQREP_SIGNAL

(SIGNAL_TIME, SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN, SIGNAL_STATE )

values (CURRENT TIMESTAMP, ‘CMD’, ‘CAPSTART’, ‘subname’, ‘P’);

-- LEGEND:
```

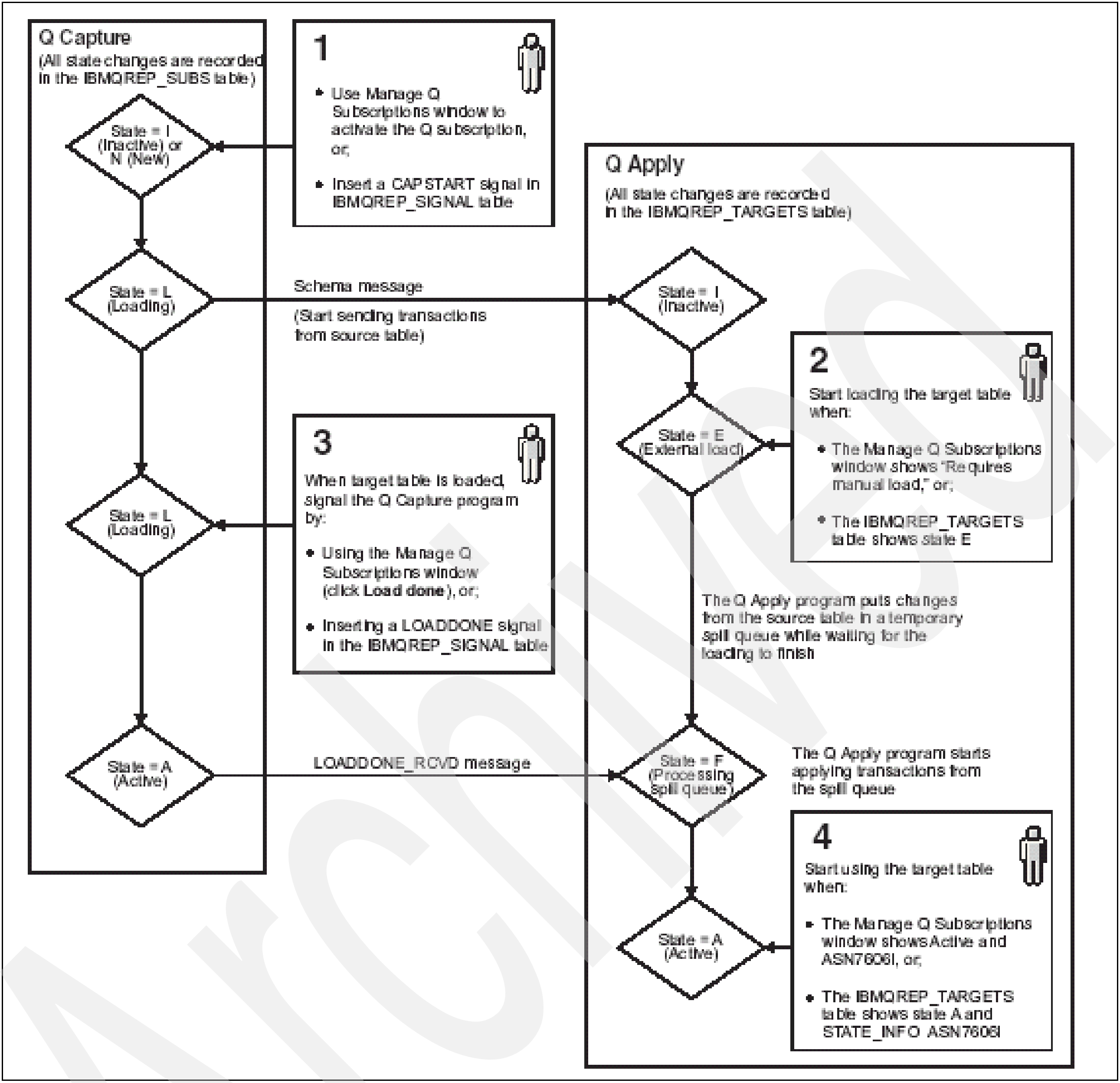
```
-- “schema” identifies the Q Capture program that you want to signal
-- “subname” is the name of the Q subscription
```



Name	System	Instance	Database	Type	Server type	Node	Comment
D8G1	9.12.6.77	WTSC53	DB8G	Remote			
DT11	9.30.132...	STLABD1	DSNT1	Remote			

Figure 2-3 Activating Q subscription via the Replication Center

Figure 2-4 describes the four main steps involved when manual loading of the target table is configured for a subscription in a unidirectional Q replication configuration.



Q Capture
(All state changes are recorded in the IBMQREP_SUBS table)
State = I (Inactive) or N (New)
State = L (Loading)
State = L (Loading)
State = A (Active)
1
Use Manage Q Subscriptions window to activate the Q subscription, or;
Insert a CAPSTART signal in IBMQREP_SIGNAL table
Schema message
(Start sending transactions from source table)
3
When target table is loaded, signal the Q Capture program by:
Using the Manage Q Subscriptions window (click Load done), or;
Inserting a LOADDONE signal in the IBMQREP_SIGNAL table
LOADDONE_RCVD message
Q Apply
(All state changes are recorded in the IBMQREP_TARGETS table)
State = I (Inactive)
State = E (External load)
2
Start loading the target table when:
The Manage Q Subscriptions window shows "Requires manual load," or;
The IBMQREP_TARGETS table shows state E
The Q Apply program puts changes from the source table in a temporary spill queue while waiting for the loading to finish
State = F (Processing spill queue)
The Q Apply program starts applying transactions from the spill queue
4
Start using the target table when:
The Manage Q Subscriptions window shows Active and ASN7606I, or;
The IBMQREP_TARGETS table shows state A and STATE_INFO ASN7606I
State = A (Active)

Figure 2-4 Manual load processing flow in unidirectional Q replication

The manual load process is described in further detail as follows:

1. When a Q subscription is activated, the Q Capture program sends a subscription "schema" message indicating that the target table will be manually loaded. The following changes occur:
   a. The Q Capture program changes the Q subscription state from "I" (inactive) or "N" (new) to "L" (loading) in the IBMQREP_SUBS control table.
   b. The Q Apply program changes the Q subscription state from "I" (inactive) to "E" (external load) in the IBMQREP_TARGETS control table.
   c. The Manage Q Subscriptions window in the Replication Center shows the state as `Requires manual load`. To open the window, right-click the Q Capture server where the source table for the Q subscription is located and select **Manage → Q Subscriptions**.

   d. The Q Apply program drops any referential integrity constraints that are defined on the target table, but it saves these constraint definitions in the IBMQREP_SAVERI table for reinstatement at the end of the load.
2. After the Manage Q Subscriptions window shows `Requires manual load` or if a SELECT statement against the IBMQREP_TARGETS table verifies that the value in the STATE column is "E", one can start loading the target table with a utility of choice.
3. While the target table is being loaded, the Q Capture program sends transactions from the source table with both before and after values. The Q Apply program puts these transactions in a temporary spill queue.
4. After the target table is loaded, one needs to notify the Q Capture program that the load is complete. You can use one of the following methods:
   – Use the Manage Q Subscriptions window in the Replication Center to indicate that the load is done.

Insert a LOADDONE signal into the IBMQREP_SIGNAL table, as shown in Example 2-2.

Example 2-2 Inserting a row in the IBMQREP_SIGNAL table

```
insert into capture_schema.IBMQREP_SIGNAL

(SIGNAL_TIME, SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN, SIGNAL_STATE )

values (CURRENT TIMESTAMP, ‘CMD’, ‘LOADDONE’, ‘subname’, ‘P’);

-- LEGEND:

-- “schema” identifies the Q Capture program that you want to signal

-- “subname” is the name of the Q subscription for which manual load is being
-- performed.
```

5. After the Q Capture program is notified that a manual load is complete, it changes the state of the Q subscription to "A" (active) in the IBMQREP_SUBS table, and begins using the sending options that are defined for the Q subscription. The Q Capture program sends a load done received message to the Q Apply program.
6. The Q Apply program changes the state of the Q subscription to "F" (processing spill queue) and starts applying transactions from the spill queue. When the Q Apply program is finished, it deletes the spill queue.
7. The Q Apply program waits until any dependent Q subscriptions have completed their load phase before putting referential integrity constraints back on the target table.
8. The Q Apply program changes the Q subscription state to "A" (active) and the STATE_INFO column to "ASN7606I" in the IBMQREP_TARGETS table and begins applying transactions from the receive queue. The Manage Q Subscriptions window shows the state as `Active`.

Important: It is the user's responsibility to ensure that applications do not access or update the target tables until they have been synchronized with the source, by checking the active state and STATE_INFO ("ASN7606I") to avoid accessing inconsistent data.

Attention: If automatic loading is chosen for a subscription, the transition from an inactive or new state for a subscription to the active state occurs without any user intervention. A shown in Figure 2-4 on page 13, this would eliminate the human intervention of the third step, which is the notification to the Q Capture of the completion of the manual load via the Manage Q subscription window of the Replication Center or the insert of the LOADDONE signal into the IBMQREP_SIGNAL table.

The flows involved for a bidirectional or peer-to-peer Q replication environment are more complex than the unidirectional flow described, and are not covered here.

2.2.2 Ongoing replication after the initial synchronization

Figure 2-1 on page 8 describes a simple unidirectional replication configuration, highlighting the main objects involved in the process.

Figure 2-5 provides a simplistic view of the elements used in replicating changes occurring at the source to the target over WebSphere MQ queues using the Q Capture and Q Apply programs.

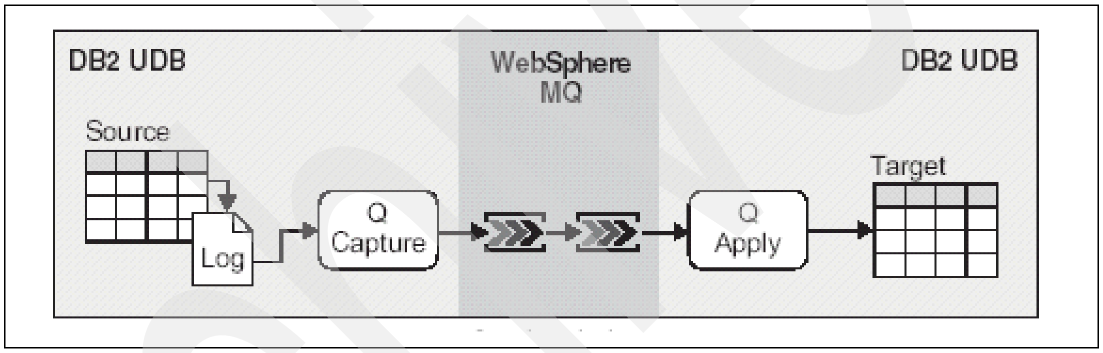


Figure 2-5 Ongoing replication flow

The process of capturing changes at the source, transporting them over WebSphere MQ to the target, and applying the changes at the target occurs in three distinct phases:

1. The first phase involves capturing changes from the log and writing them to the WebSphere MQ send queue.
2. The second phase involves WebSphere MQ transporting the messages from the source to the target.
3. The third phase involves applying the changes to the target.

A number of parameters determine the latency, which is the delay between the time transactions are committed at the source and target servers.

A brief review of the three main components involved (namely Q Capture, WebSphere MQ, and Q Apply) follows.

Q Capture

The Q Capture program is a program that reads the DB2 recovery log for changes that occur in source tables, turns the changes into messages, and sends the messages over WebSphere MQ queues, where the messages are processed by a Q Apply program or user application.

As soon as a subscription is activated, Q Capture begins collecting changes for the source table identified in the subscription and writes them out to a WebSphere MQ queue at the transaction boundary.

The Q Capture program lets one specify the data to be published or replicated from the source table:

- One can control which source changes are sent by specifying the source tables, or even rows and columns within the source tables.
- One can control the latency and amount of data that flows over queues by setting the (commit) interval for which the Q Capture program commits messages, among many other parameters.

Note: Multiple Q Capture programs may be configured to independently capture data on a single source server. Each Q Capture program is identified on a server by a unique schema name. (A schema is the name given to the set of control tables used by a Q Capture or Q Apply program. Q replication uses schemas to identify the Q Capture or Q Apply program that uses a specific set of control tables.)

When a row changes in a source table, the Q Capture program reads the log record to see if the table is part of an active Q subscription or XML publication. If so, a Q Capture program adds the row to the corresponding database transaction in memory until it reads the corresponding commit or abort record from the database log.

- If a row change involves columns with large object (LOB) data, the Q Capture program copies the LOB data directly from the source table to the send queue.
- If one defines a search condition, the Q Capture program uses it to evaluate each row in memory. Rows that meet the search condition are assembled into messages when the transaction that they belong to is committed at the source database.

The Q Capture program then puts the assembled messages on all send queues that were defined for the Q subscriptions.

One can modify Q Capture parameters to reduce the latency time and minimize network traffic by filtering rows with a search condition, limiting which column values are sent, or not propagating deletes. One can specify a search condition for each Q subscription or XML publication. The search condition is a restricted SQL WHERE clause. Row changes that do not meet the search condition are not included in messages. Using a search condition is a trade-off between CPU consumption and latency. A dedicated log reader thread reads the log records, while a worker thread evaluates the predicate.

Note: For XML publications, one can limit which column values are added to messages by choosing from the following options. (For Q subscriptions, these column options are selected automatically based on the selected conflict options.)

- All changed rows: By default, the Q Capture program sends a row only when a column that is part of a Q subscription or XML publication changes. One can choose to have the program send a row when *any* column changes.
- Before values: By default, the Q Capture program does not send before values of non-key columns that are updated. One can choose to have the program send before values of non-key columns.
- Changed columns only: By default, the Q Capture program sends only subscribed columns that have changed. You can choose to have the program also send subscribed columns that did not change.

A subscription's default is to have the Q Capture program capture deletes from the log and publish or replicate them. One can, however, choose to have deletes suppressed for a given subscription.

A Q Capture program can translate changes from a source table into two different message formats, as follows:

1. A compact format that the Q Apply program can read.
2. An XML format that the event publisher uses or can be used by a user application. One can choose whether messages contain a single row operation only, or all the row operations within a transaction.

After the Q Capture program puts messages on one or more WebSphere MQ send queues, it issues a commit call to the WebSphere MQ queue manager instructing it to make the messages on the send queues available to the Q Apply program or user applications. One can configure how often the Q Capture program commits messages. All of the DB2 transactions grouped within each commit are considered to be a single WebSphere MQ transaction. Typically, each WebSphere MQ transaction contains several DB2 transactions. One can adjust the time between commits by changing the value of the COMMIT_INTERVAL parameter of Q Capture. A shorter commit interval can lower end-to-end latency up to an equilibrium point—500ms, which is the default (optimal result from performance studies).

The Q Capture operating parameters govern how much memory the Q Capture program allocates for building transactions, the actions that it takes when starting, how often it deletes old data from its control tables, and other behaviors. These parameters can be changed in three ways, as follows:

- By updating the control table where the Q Capture program reads its parameter values
- By temporarily overriding the saved values when you start the program
- By changing the parameters dynamically while the program is running

Attention: With the latter two methods, the changes last only while the Q Capture program is running. When it stops and restarts, the Q Capture program uses the values that are saved in the control table, unless someone overrides the values again.

A Q Capture program responds to commands, SQL signals, and XML and compact messages as follows:

1. The Replication Center or system commands may be used to control the following behaviors of the Q Capture program:
   - Start a Q Capture program and optionally change startup parameters.
   - Change parameter values while a Q Capture program is running.
   - Re-initialize a Q Capture program.
   - Re-initialize a send queue.
   - Stop a Q Capture program.
2. The Replication Center issues SQL signals to communicate the following requests to a Q Capture program—one may also manually insert SQL signals into the IBMQREP_SIGNAL table to perform these tasks.
   - Request that the Q Capture program activate or deactivate a Q subscription or XML publication.
   - Report that a target table is loaded.
   - Tell the Q Capture program to re-initialize a Q subscription or XML publication.

One can manually insert SQL signals into the IBMQREP_SIGNAL table to perform the following additional tasks:

– Add a column to an active unidirectional Q subscription or XML publication.
– Execute the error action that is defined for a send queue.
– Stop the Q Capture program.
– Ignore a transaction.

3. The Q Apply program and user applications communicate with the Q Capture program by sending compact and XML messages, respectively.

The Q Capture program uses a local WebSphere MQ queue to store restart information. The restart queue contains a single message that tells the Q Capture program where to start reading in the DB2 recovery log when the Q Capture program restarts. Each time that the Q Capture program reaches its commit interval, it checks to see whether it needs to update its restart information. If so, the Q Capture program replaces the message on the restart queue with a new message that contains relevant restart information including, among other things, the earliest point in the log at which it needs to start processing log records upon restart. When the cold start option is used, the Q Capture program replaces the restart message with a message that indicates for the program to start processing log records at the current point in the log.

The Q Capture program can be controlled by a number of parameters including the amount of memory that can be used to build transactions, the commit interval, and the monitor interval that specifies how often the Q Capture program writes to the trace tables. For details on these parameters, refer to the DB2 Information Center:

http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp

WebSphere MQ

WebSphere MQ ensures that every message[2] generated by the Q Capture program is transported to the target without loss and in the correct order. Since the WebSphere MQ queues used by replication must be defined as being persistent, the messages in the queues can survive crashes.

Depending on the type of replication or publishing to be performed, various WebSphere MQ objects are required. The following is a summary of the WebSphere MQ objects required by the Q Capture and Q Apply programs, with a brief description of their usage.

- *Queue manager* is a program that manages queues for Q Capture programs, Q Apply programs, and user applications. One queue manager is required on each system.
- *Send queue* is a queue that directs data, control and informational messages from a Q Capture program to a Q Apply program or user application. In remote configurations, this is defined as a remote queue on the source system corresponding to the receive queue on the target system. Each send queue should be used by only one Q Capture program.
- *Receive queue* is a queue that receives data and informational messages from a Q Capture program to a Q Apply program. This is a local queue on the target system.
- *Administration queue* is a queue that receives control messages from a Q Apply program or a user application to the Q Capture program. This is a local queue on the system where the Q Capture instance runs. There is a remote queue definition on the system where the Q Apply program or a user application runs, corresponding to the administration queue where the Q Capture instance runs.

[2] Q replication messages are persistent.

- *Restart queue* is a queue that holds a single message that tells the Q Capture program where to start reading in the DB2 recovery log after a restart. This is a local queue on the source system. Each Q Capture program must have its own restart queue.
- *Spill queue* is a model queue that one defines on the target system to hold transaction messages from a Q Capture program while a target table is being loaded. The Q Apply program creates these dynamic queues during the loading process based on the model queue definition, and then deletes them. A spill queue (with any user-defined name) may be specified at the subscription level.

The WebSphere MQ objects must be defined for Q replication or event publishing.

Note: While the definitions of the WebSphere MQ objects need to be synchronized in WebSphere MQ and Q replication or event publishing, the sequence in which these are defined is immaterial. However, the IBM manuals recommend defining the objects in WebSphere MQ first, before defining them in Q replication or event publishing to take advantage of the facility in the Replication Center to select WebSphere MQ objects from a list to minimize typing errors.

The WebSphere MQ objects are defined in Q replication or event publishing when the Q Capture and Q Apply tables are defined, and when the replication queue maps and publishing queue maps are defined.

- For the Q Capture control tables, you must provide the name of a queue manager on the system where the Q Capture program runs, and the name of a local administration queue and local restart queue.
- For the Q Apply control tables, you must provide the name of a queue manager on the system where the Q Apply program runs.
- For the replication queue maps, you must provide the name of a send queue on the system where the Q Capture program runs, and a receive queue where the Q Apply program runs. An administration queue must also be defined for a replication queue map that the Q Apply program uses to send control messages to the Q Capture program.
- For the publishing queue maps, you must provide the name of a send queue on the system where the Q Capture program runs.

Note: A WebSphere MQ client may be defined that allows WebSphere MQ objects used by Q replication to be on a different server from the ones where Q Capture and Q Apply run. However, we recommend a local queue manager for performance reasons.

Note: The WebSphere MQ transmission queues and channels do not have to be defined for the Q replication or event publishing programs. They only need to be defined within the source and target queue managers.

WebSphere MQ objects allow multiple settings to control their behavior. The queue manager, for instance, allows one to limit the maximum size (MAXMSGL parameter) of the messages on the queue, while the queues allow one to specify the maximum number of messages (MAXDEPTH) in the queue and whether they are to be persistent or shared. The channels allow one to set the disconnect interval that specifies the duration the channel is to remain open when there are no transactions to replicate.

Q Apply

The Q Apply program takes messages from WebSphere MQ queues, rebuilds the transactions that the messages contain, and applies the transactions to target tables or stored procedures as the case may be.

The Q Apply program is designed to keep up with rapid changes at multiple sources by applying transactions to multiple targets at the same time. The program can apply changes in parallel based on a dependency analysis, while maintaining referential integrity between related target tables.

Note: Multiple Q Apply programs may be configured to independently apply data to a single target server. Each Q Apply program is identified on a server by a unique schema name. (A schema is the name given to the set of control tables used by a Q Capture or Q Apply program. Q replication uses schemas to identify the Q Capture or Q Apply program that uses a specific set of control tables.)

To allow the Q Apply program to track what changes it has already applied to the target, each target table must have some mechanism to enforce that rows are unique. This uniqueness can come from a primary key, unique constraint, or unique index at the target. One can specify whether the program applies transactions in parallel, change its operating behavior in several ways, and set options for how the Q Apply program detects and handles conflicts[3].

Note: As of fix pack 10, you can now replicate to target tables, even if your source table has no primary key or unique constraint. If you are replicating large object (LOB) values, the source table still must have a unique constraint. This constraint is used to identify the LOB values, which are fetched directly from the source table.

A Q Apply program can handle transactions from multiple receive queues. For each receive queue, the Q Apply program launches a single thread known as a browser thread. The browser thread gets transaction messages from the receive queue and keeps track of dependencies between transactions. The browser thread also tracks which transactions it has already applied.

Note: To maintain referential integrity between dependent transactions, each replication queue map (which identifies a send and receive queue pair that is associated with one or more subscriptions), which identifies the receive queue, must be processed by a single browser thread, and Q subscriptions that involve dependent tables must use the same replication queue map. Therefore, two Q Apply programs cannot get transactions from the same replication queue map.

Each browser thread launches one or more agent threads. The agent thread takes transaction messages from the browser thread, rebuilds the SQL statements for all row changes, applies the changes to targets, and issues the commit statement for transactions.

WebSphere MQ ensures that transaction messages arrive in a receive queue in the same order[4] that they were committed at the source. By default, the Q Apply program applies

[3] When a change is replicated to the target and that row (column) at the target has been modified by an application, thereby resulting in a mismatched column value, missing row, or duplicate row condition in Q Apply

[4] This statement is true only if the message delivery sequence FIFO is defined on the transmission queue (the default), and also when there is no Dead Letter Queue (DLQ) defined. If a DLQ is defined, it is possible for some messages to be routed there. These messages may then be picked up by the DLQ handler and returned to the original queue, but be “out of order” with reference to sending. The Q replication Apply browser thread requests messages according to a sequence number and does not rely on them being physically in sequence in the receive queue.

transactions in parallel using multiple agent threads. Such parallelism is important when many changes are replicated from the source server.

Note: One can set the number of agent threads to one to ensure that the Q applies transactions in their strict arrival order.

The Q Apply program takes special precautions when it applies transactions in parallel, since rows are not always applied in the same order in which the changes occurred at the source. Changes to the same row at the target are serialized according to the modification sequence at the source; the same applies to parent/child updates as well.

For example, assume that two subscriptions have been defined—one on the DEPARTMENTS table, and the other on the EMPLOYEE table—and that a referential constraint exists between the two tables. Since the subscriptions are related, they share the same replication queue map and therefore the same receive queue. Assume a new department is created by inserting a row into the DEPARTMENTS table at the source, followed by an insert of an employee to the new department into the EMPLOYEES table at the source. The referential integrity constraint at the source requires that any row inserted into the EMPLOYEES table must have a matching record in the DEPARTMENTS table. The same parent-child dependency exists between the replicated copies of these two tables. When messages that contain these source transactions arrive on the receive queue, the browser thread detects the dependency between these two transactions. If the Q Apply program is using multiple agents to apply transactions in parallel, the browser thread still performs the inserts in parallel. If the insert of the dependent row fails with a -530 SQL code, Q Apply just retries the insert. This approach is optimistic in that it enforces serialization while enhancing performance through parallelism. Meanwhile, agent threads continue to apply other transactions in parallel.

The Q Apply operating parameters let one set how often the Q Apply program saves or prunes performance data, where it stores its diagnostic log, and how often it retries to apply changes to targets after deadlocks or lock time-outs. These parameters can be changed in three ways, as follows:

- By updating the control table where the Q Apply program reads its parameter values
- By temporarily overriding the saved values when you start the program
- By changing the parameters dynamically while the program is running

Attention: With the latter two methods, the changes last only while the Q Apply program is running. When it stops and restarts, the Q Apply program uses the values that are saved in the control table, unless one overrides the values again.

The number of apply agents and the memory limit parameters determine the Q Apply program's behavior for each receive queue that it works with. One can specify the value of these parameters when creating the replication queue map. These parameter values can be changed without stopping the Q Apply program.

The Q Apply program can be controlled by a number of parameters, including the monitoring interval that specifies how often Q Apply writes monitor information, and the pruning interval for emptying rows in the trace tables.

2.3 Choosing a particular Q replication topology

Table 2-1 provides a high-level overview of some of the key criteria involved in choosing a particular Q replication topology to address a business requirement.

Table 2-1 Choosing a particular Q replication topology

Evaluation criteria	Unidirectional	Bidirectional	Peer-to-peer
One-way replication	Yes		
Subset rows and columns in target	Yes		
Data transformation required	Yes		
"Toggle"[a] replication between two servers		Yes	
More than two servers involved in replication			Yes
Conflict detection and resolution requirements: ► "Master" wins the conflict. ► Not detecting all conflicts is acceptable (such as LOB conflicts). ► The latest update wins the conflict. ► Conflict detection is in LOB values. ► Adding columns and triggers to the replicated tables for conflict detection and resolution is acceptable.		 Yes Yes Yes	 Yes Yes Yes Yes
High availability usage scenario requirement: ► Fast takeover in the event of failure		Yes	Yes Yes

a. In "toggle" replication, one server can be updated while the other is read-only until failover.

When multiple topologies can address the business requirement, it would probably be appropriate to choose the topology that is least expensive and has minimal impact on existing data. For example, if either bidirectional or peer-to-peer replication would address the business requirement, it would be appropriate to choose bidirectional since it does not require the source table to have additional columns and triggers defined (unlike the case with peer-to-peer replication), and the overhead of conflict detection and resolution is lower than that of peer-to-peer replication. However, conflict detection and resolution should also be a determining factor, as peer-to-peer replication conflict detection and resolution is more robust than that of bidirectional replication.

2.4 Latency considerations

As mentioned earlier, latency is a measure of the time it takes for transactions to replicate from the Q Capture server to the Q Apply server. The Q Capture and Q Apply programs save performance data that lets one track latency between various stages of the replication process. These statistics can help one pinpoint problems and tune one's environment. Figure 2-6 shows the various types of latency defined in Q replication, while Figure 2-7 shows how these statistics may be computed using SQL.

1 Time that the transaction committed
2 Time that Q Capture read the transaction from the log
3 Time that Q Capture put the message to the Sendqueue
4 Time that Q Apply received the message from the Recvqueue
5 Time that Q Apply copied the transaction to the target tables

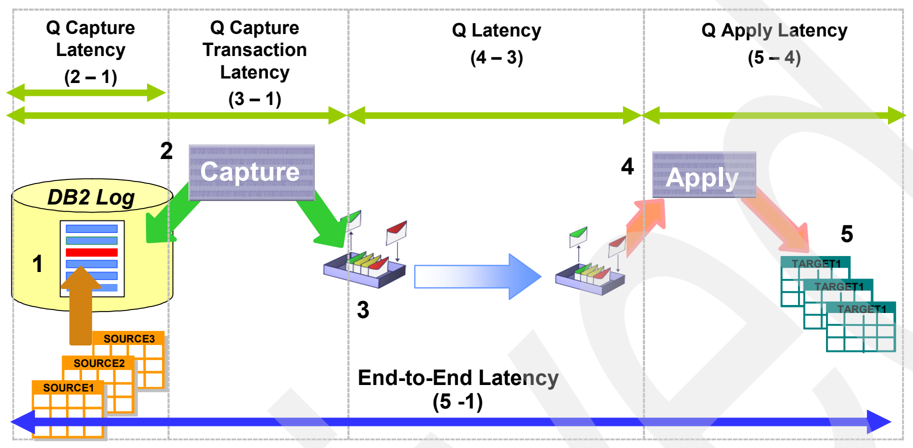


Figure 2-6 Latency statistics collected

Latency	Control tables
Q Capture Latency	IBMQREP_CAPMON control table (MONITOR_TIME - CURRENT_LOG_TIME)
Q Capture transaction Latency	IBMQREP_APPLYMON control table (END2END_LATENCY – QLATENCY - APPLY_LATENCY)
Queue Latency	IBMQREP_APPLYMON control table QLATENCY column
Q Apply Latency	IBMQREP_APPLYMON control table APPLY_LATENCY column
End-to-End Latency	IBMQREP_APPLYMON control table END2END_LATENCY column

Figure 2-7 Computing latency statistics

By using the Latency window in the Replication Center, one can determine how long it takes for transactions to move between the:

- DB2 recovery log and the send queue (Q Capture transaction latency)
- Q Capture program putting a transaction on a send queue and the Q Apply program getting the transaction from the receive queue (Q latency)
- Receive queue and the target table (Q Apply latency)

One can also use the Q Capture Latency window to view an approximate measure of how a Q Capture program is keeping up with changes to the log (Q Capture latency).

Attention: Any latency measure that involves transactions that are replicated between remote Q Capture and Q Apply servers can be affected by clock differences between the source and target systems. To get a true measure, ensure that the clocks are synchronized. Also, achieving good latency requires the Q Capture and Q Apply programs to run continuously.

Each of these measures is discussed in further detail in the following sections.

2.4.1 Q Capture latency

Q Capture latency measures the difference between a given point in time and the timestamp of the last committed transaction that Q Capture read. This measure uses the values of the MONITOR_TIME and CURRENT_LOG_TIME columns in the IBMQREP_CAPMON control table. When examined in aggregate, these latency measurements can help you determine how well a Q Capture program is keeping up with the database log.

For example, if a Q Capture program inserted a row of performance data into the IBMQREP_CAPMON table (MONITOR_TIME) at 10 a.m. and the timestamp of the last committed transaction (CURRENT_LOG_TIME) is 9:59 a.m., then the Q Capture latency would be one minute.

If the Q Capture latency is considered too high, log-reading performance may be improved by creating an additional Q Capture schema and moving some Q subscriptions or XML publications to the new schema. Each additional schema has its own transaction thread to read the log.

Note: In a z/OS data sharing environment, an additional Q Capture schema may add to CPU consumption as well as latency.

2.4.2 Q Capture transaction latency

Q Capture transaction latency measures the time between the Q Capture program reading the commit statement for a transaction in the DB2 recovery log, and the message containing the transaction being put on a send queue. This statistic provides information about how long it takes the Q Capture program to reassemble transactions in memory, filter out rows and columns based on settings for the Q subscription or XML publication, and then put the transaction messages on a queue.

If this latency is considered too high, it may be reduced by:

- Increasing the value of the MEMORY_LIMIT parameter, which sets the total amount of memory allocated by a Q Capture program. This only helps with large transactions that spill to disk during Q Capture.

  Note: In an upcoming PTF PK14999, the recommendation will be to set the MEMORY_LIMIT to zero and let Q Capture optimize the MEMORY_LIMIT from the region size.

- Raising the MAX_MESSAGE_SIZE parameter, which is defined when you create a replication queue map or publication queue map. This parameter sets the amount of memory that a Q Capture program allocates for building transaction messages for each send queue. If the maximum message size is too small, the Q Capture program divides transactions into multiple messages, requiring more processing time and increasing latency.

- Reducing the COMMIT_INTERVAL.

2.4.3 Queue latency

Q latency measures the time between the Q Capture program putting a transaction on a send queue and the Q Apply program getting the transaction from the receive queue. This statistic provides information about WebSphere MQ performance.

Please refer to the *WebSphere MQ, System Administration Guide*, SC34-6069-01, for tuning this environment.

2.4.4 Q Apply latency

Q Apply latency measures the time it takes for a transaction to be applied to a target table after the Q Apply program gets the transaction from a receive queue. The more agent threads that you have specified for a receive queue, the smaller this number should be.

Note: When the Q Apply program delays applying a transaction involving dependent tables until all previous transactions that it depends on have been applied, it results in an increase in Q Apply latency.

If this latency is considered too high, it may be reduced by:

- Increasing the NUM_APPLY_AGENTS if it is determined to be due to an under-configured value
- Increasing the MEMORY_LIMIT if it is determined to be due to insufficient memory to perform parallel operations
- Tuning the database to reduce deadlocks that can contribute to increased apply latency

2.4.5 End-to-end latency

End-to-end latency measures the time between the Q Capture program reading the log record for a transaction commit statement and the Q Apply program committing the transaction at the target. This statistic is an overall measure of Q replication latency and an aggregation of the individual components described.

If this latency is considered too high, it may be reduced by addressing the individual components that make up this latency, namely, Q Capture transaction latency, Q latency, and Q Apply latency.

3

Failover and switchback scenarios

In this chapter we provide a step-by-step approach to performing failover and switchback in a bidirectional replication scenario involving two z/OS servers.

The topics covered are:

- Business requirement
- Rationale for the bidirectional solution
- Environment configuration
- Failover and switchback considerations
- Controlled failover and switchback
- Uncontrolled failover and switchback

3.1 Introduction

Bidirectional replication using Q replication may be the replication of choice for environments that require a high-volume, low-latency solution between one or more tables on two servers, with a capability to update tables on both servers with certain restrictions. Scenarios that may be appropriate for a bidirectional replication solution include the following:

- There is little or no potential for the same data in the replicated tables to be updated simultaneously.

  In this scenario, different rows of a table may be updated at the two servers. An example is where each server acts as a master for a particular region—serverA only has updates to the western region data while serverB only has updates for the eastern region.
- The second server is maintained as a hot site back up and is not updated (other than through replication) while the first server is available. When the first server fails, applications are “switched over” to use the second server, which then allows updates to occur.

  In this scenario, the second server tables may or may not be made available for read-only access while the primary is still available.

In this chapter we describe a high availability business solution implementation involving bidirectional Q replication. It starts with a definition of the business requirements, then a selection of the appropriate technology solution to address it, and finally implementing it in the target environment. The entire process and associated considerations is documented as follows:

- Business requirement
- Rationale for choosing the bidirectional solution
- Environment configuration
- Failover and switchback considerations
- Controlled failover and switchback
- Uncontrolled failover and switchback
- Switchback considerations

3.2 Business requirement

Our fictitious company ABC Financials maintains financial data for a government organization and is contracted to maintain a non-dedicated hot-site backup (not disaster recovery) to which applications can be switched within minutes in the event of a failure of the primary server or extended maintenance to the primary server. The secondary server is meant to be a temporary alternative until the primary server is restored to full service. In addition, ABC Financials is required to make the hot site available for read-only access for reporting purposes to offload processing from the primary server.

These requirements may be summarized as follows:

1. Maintain a non-dedicated hot-site backup. This means that the hot-site backup server has actively updated application tables unrelated to the financial data application.
2. Replicated tables on the second server must be available on failover within 30 minutes.
3. Replicated tables on the second server must be available for read-only access while the primary server is available.

4. Primary server to be re-synchronized with the secondary after it is restored to service.
5. Limited resources allowed for replication-only purposes.

3.3 Rationale for the bidirectional solution

Based on the considerations discussed in Section 2.3, “Choosing a particular Q replication topology” on page 22, and the business requirements defined in Section 3.2, “Business requirement” on page 28, the bidirectional replication topology is appropriate for ABC Financials for the following reasons:

1. Less stringent requirements for failover time (within 30 minutes instead of an instantaneous requirement that would tilt in favor of peer-to-peer).
2. Non-dedicated backup server requirement.
3. Absence of conflicts[1] due to the lack of simultaneous updates. Only one server is updated at a given time.
4. As indicated earlier, when multiple topologies can address a business requirement, it would probably be appropriate to choose the topology that is least expensive and has the minimal impact on existing data. Bidirectional is the lower-cost alternative and does not require additional columns to be defined on existing data.
5. ABC Financials’ requirement for a controlled switchback to the primary server from the secondary server is handled well by a bidirectional replication topology.

3.4 Environment configuration

Figure 3-1 shows the configuration used in the ABC Financials’ bidirectional replication topology.

[1] During switchback, there is likely to be some data loss and conflicts due to the fact that all the changes on the primary server at the time of failure fail to get replicated over to the secondary server. These condition are resolved partially by the conflict resolution mechanism during switchback and may require manual intervention.

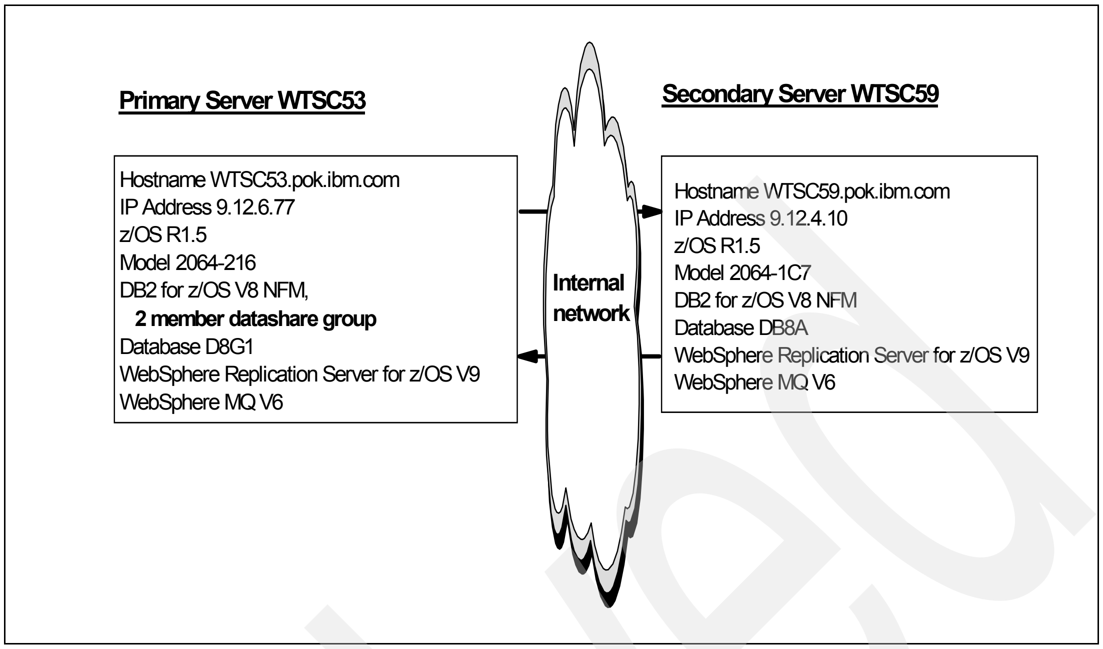


Figure 3-1 Bidirectional replication topology configuration

We installed a set of four tables with referential integrity constraints defined between some of them, as shown in Figure 3-2. The combination of delete RESTRICT and delete CASCADE options were essential to describing the conflict and SQL error exceptions. The OPTIONS_V1 table has a CLOB column.

Note: The identity column in the TRAN_V1 table on the secondary server starts with 2 with increments by 2. This ensures that the identity column values generated on the secondary server is always even, while the values generated for the corresponding column on the primary server is always odd. We strongly recommend this approach to tables with identity columns involved in bidirectional (and peer-to-peer) replication implementations to avoid data corruption in conflict situations. This is described later.

The DDL used in creating these tables is shown in Example 3-1.

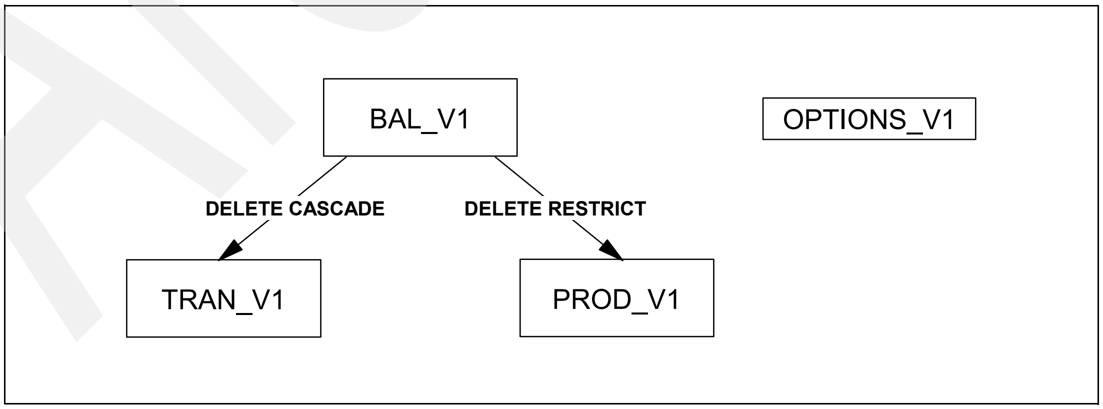


Figure 3-2 Tables used in the bidirectional replication scenario

Example 3-1 DDL of tables used in the bidirectional replication scenario

```
------------------------------------------------
-- DDL Statements for table ITSO.PROD_V1UCT
------------------------------------------------

CREATE TABLE ITSO.PROD_V1  (
  ACCT_ID INTEGER NOT NULL
, PROD_DATE DATE NOT NULL
, PROD_TYPE CHAR(10) NOT NULL WITH DEFAULT
, PROD_QTY  SMALLINT  NOT NULL WITH DEFAULT
, PRIMARY KEY (ACCT_ID, PROD_TYPE)
)
DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSO.PROD_V1_PK ON ITSO.PROD_V1 (ACCT_ID, PROD_TYPE);


------------------------------------------------
-- DDL Statements for table ITSO.TRAN_V1
------------------------------------------------

CREATE TABLE ITSO.TRAN_V1  (
  ACCT_ID INTEGER NOT NULL
, TRAN_ID INTEGER GENERATED BY DEFAULT AS IDENTITY
                  (START WITH 1 INCREMENT BY 2)
, TRAN_DATE DATE NOT NULL
, TRAN_TYPE CHAR(2) NOT NULL WITH DEFAULT
, TRAN_QTY  DECIMAL(15, 5) NOT NULL WITH DEFAULT
, PRIMARY KEY (ACCT_ID, TRAN_ID)
)
DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSO.TRAN_V1_PK ON ITSO.TRAN_V1 (ACCT_ID, TRAN_ID);


------------------------------------------------
-- DDL Statements for table ITSO.BAL_V1
------------------------------------------------

CREATE TABLE ITSO.BAL_V1  (
          ACCT_ID INTEGER NOT NULL
          , ACCTG_RULE_CD CHAR(1) NOT NULL
          , BAL_TYPE_CD CHAR(2) NOT NULL
                         , BAL_DATE DATE NOT NULL
                         , BAL_AMOUNT DECIMAL(15, 2) NOT NULL WITH DEFAULT
                         , PRIMARY KEY (ACCT_ID)
        )
        DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSO.BAL_V1_PK ON ITSO.BAL_V1 (ACCT_ID);

CREATE TYPE 2 UNIQUE INDEX ITSO.BAL_V1_UNQ1
ON ITSO.BAL_V1 (ACCT_ID , ACCTG_RULE_CD , BAL_TYPE_CD , BAL_DATE);


------------------------------------------------
-- DDL Statements for table ITSO.OPTIONS_V1
------------------------------------------------

create tablespace itsolobT in ITSITDB1;
create lob tablespace itsolob1 in ITSITDB1 LOG NO;

CREATE TABLE ITSO.OPTIONS_V1  (
                  OPTION_ID INTEGER NOT NULL,
        NOTES CLOB(4765) NOT NULL ,
        UPDATED DATE NOT NULL,
                  PRIMARY KEY (OPTION_ID) )
       IN ITSITDB1.ITSOLOBT
                 DATA CAPTURE CHANGES ;

CREATE TYPE 2 UNIQUE INDEX ITSO.OPTIONS_V1_PK
ON ITSO.OPTIONS_V1 (OPTION_ID);

create auxiliary table ITSO.OPTIONS_V1_aux
in itsitdb1.itsolob1 stores ITSO.OPTIONS_V1 column notes;

create unique index itso.options_v1_aux_u1
on ITSO.OPTIONS_V1_aux;


-- DDL Statements for foreign keys on Table ITSO.TRAN_V1
ALTER TABLE ITSO.TRAN_V1
   ADD CONSTRAINT FK_TRAN_V1 FOREIGN KEY (ACCT_ID)
   REFERENCES ITSO.BAL_V1 (ACCT_ID)
   ON DELETE CASCADE
   ENFORCED
   ENABLE QUERY OPTIMIZATION;

-- DDL Statements for foreign keys on Table ITSO.PROD_V1
ALTER TABLE ITSO.PROD_V1
   ADD CONSTRAINT FK_PROD_V1 FOREIGN KEY (ACCT_ID)
   REFERENCES ITSO.BAL_V1 (ACCT_ID)
   ON DELETE RESTRICT
   ENFORCED
   ENABLE QUERY OPTIMIZATION;
```

Figure 3-3 provides a high-level overview of the various objects involved in implementing a bidirectional replication topology for ABC Financials. In Figure 3-3, there appear to be two sets of transmission queues, and sender and receiver channels on each server. However, there is only one set on each server, as can be deduced from the identical names. Figure 3-3 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

Attention: The setup of the bidirectional Q replication environment described in Figure 3-1, Figure 3-2, and Figure 3-3 was performed using scripts that can be downloaded from the IBM Redbooks Web site:

ftp://www.redbooks.ibm.com/redbooks/SG247215/

Figure 3-4 provides an overview of the steps involved in setting up this environment. No further discussion of installation and configuration is presented here, since the objective of this book is to focus on high availability aspects of bidirectional replication rather than on installation and configuration, which is covered in great detail in an earlier IBM Redbook *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487.

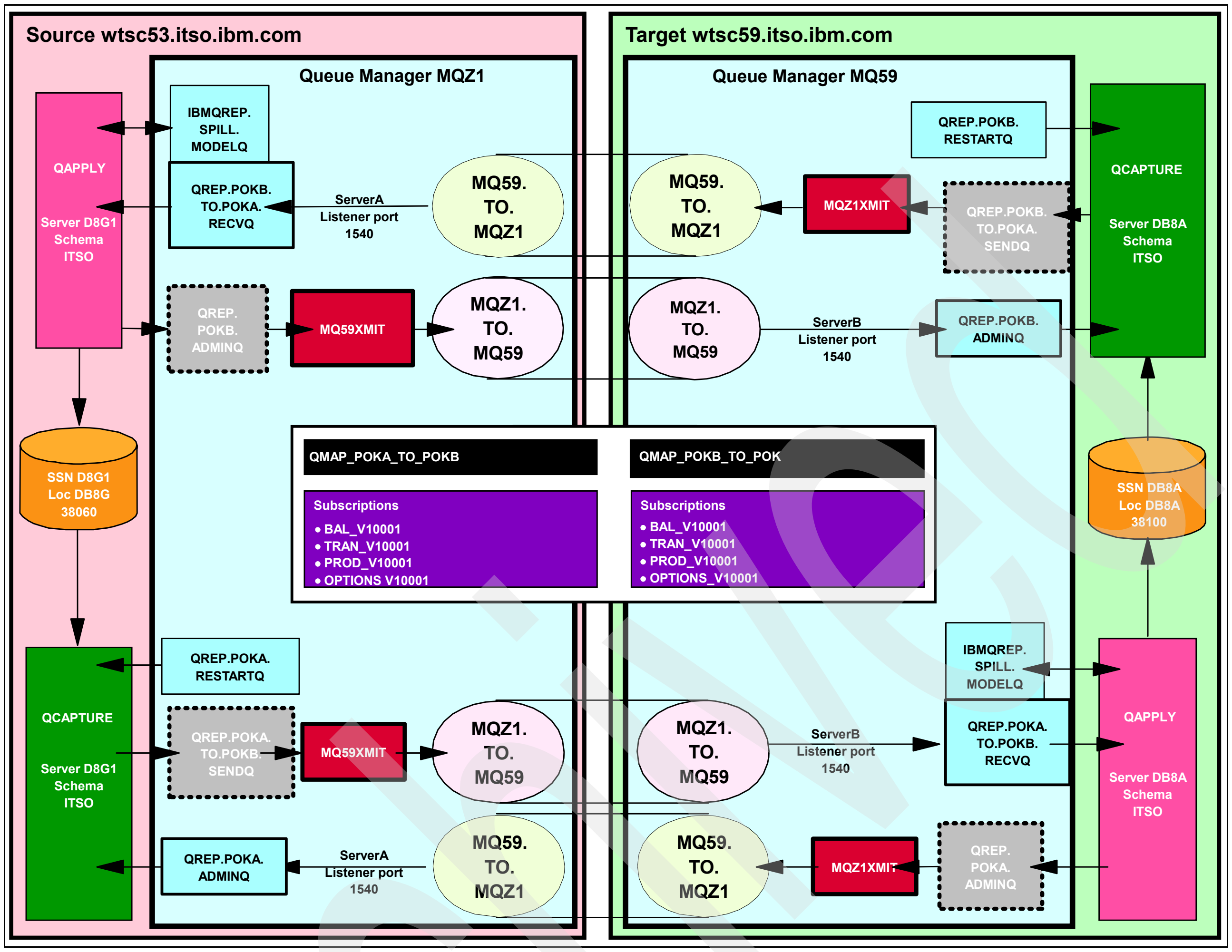


Figure 3-3 Bidirectional replication topology objects overview

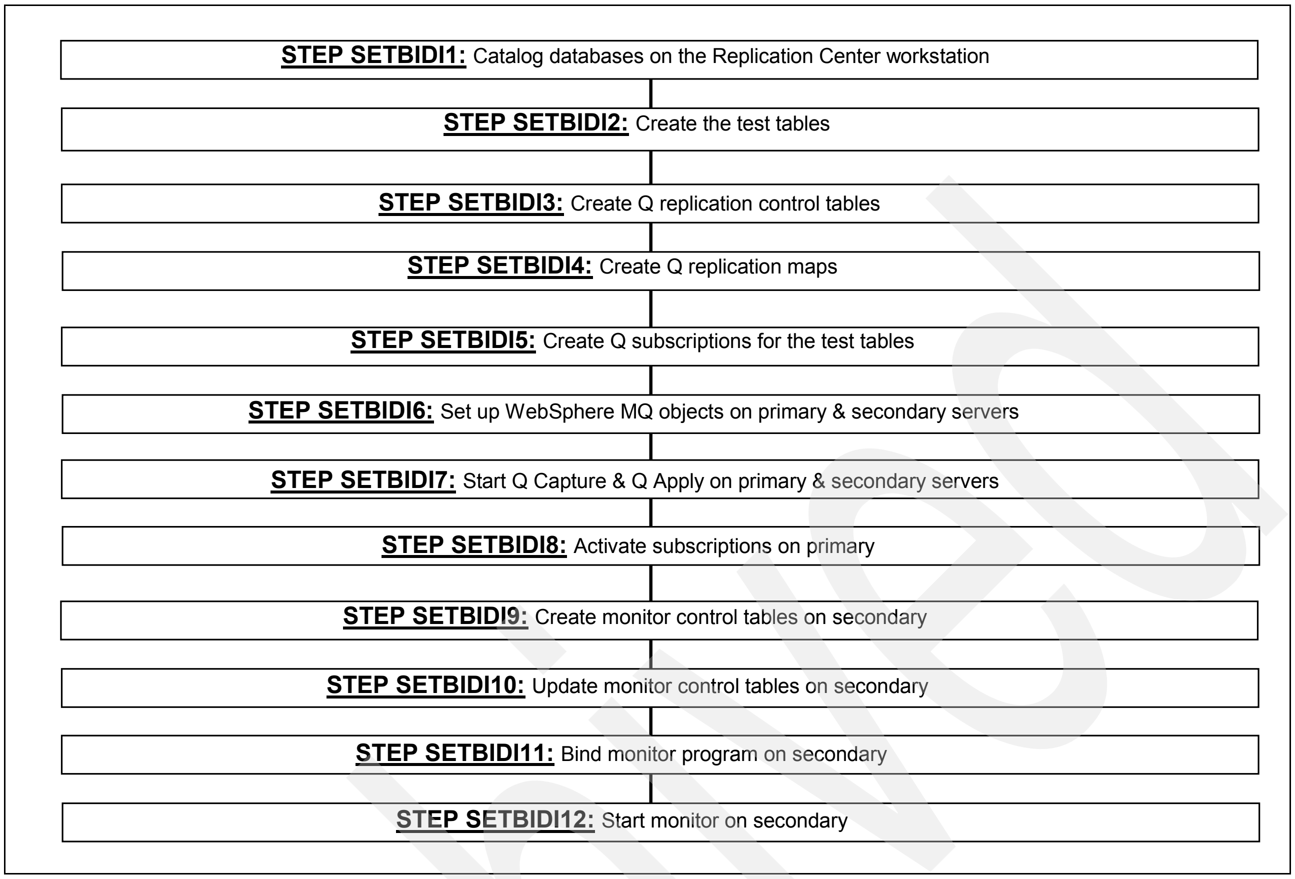

STEP SETBIDI1: Catalog databases on the Replication Center workstation
STEP SETBIDI2: Create the test tables
STEP SETBIDI3: Create Q replication control tables
STEP SETBIDI4: Create Q replication maps
STEP SETBIDI5: Create Q subscriptions for the test tables
STEP SETBIDI6: Set up WebSphere MQ objects on primary & secondary servers
STEP SETBIDI7: Start Q Capture & Q Apply on primary & secondary servers
STEP SETBIDI8: Activate subscriptions on primary
STEP SETBIDI9: Create monitor control tables on secondary
STEP SETBIDI10: Update monitor control tables on secondary
STEP SETBIDI11: Bind monitor program on secondary
STEP SETBIDI12: Start monitor on secondary

Figure 3-4 Overview of setup steps for the bidirectional replication environment

3.5 Failover and switchback considerations

This section provides a high-level overview of some of the considerations involved with failover and switchback processing associated with bidirectional replication. Depending upon the particular environment, the process involved in ensuring satisfactory failover and switchback processing can get quite complex. The topics covered here are:

- Failover processing considerations
- Switchback processing considerations
- Considerations in choosing a particular switchback scenario

3.5.1 Failover processing considerations

When the failover occurs to the secondary server, it is possible for some of the changes that occurred at the primary server not to be replicated over to the secondary server. These changes may include changes in the DB2 log that had not as yet been sent to the WebSphere MQ queue (item 1 in Figure 3-5), or messages in the WebSphere MQ queue that had not been transmitted to the secondary server (item 2 in Figure 3-5). These un-replicated changes should be considered to be “data loss” at the secondary server, *at least until the primary server is restored*.

Note: Data loss refers to transactions that are not propagated and applied to the "target" table. (Here "target" does not refer to the location of the table but the object of the replication). Therefore, changes may have originated on the primary server prior to failover, or originated on the secondary server after failover and prior to switchback.

If there are messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 3-5), when the secondary server is enabled for updates, then conflicts may occur on the secondary server between the updates in its receive queue and the updates occurring on the secondary server.

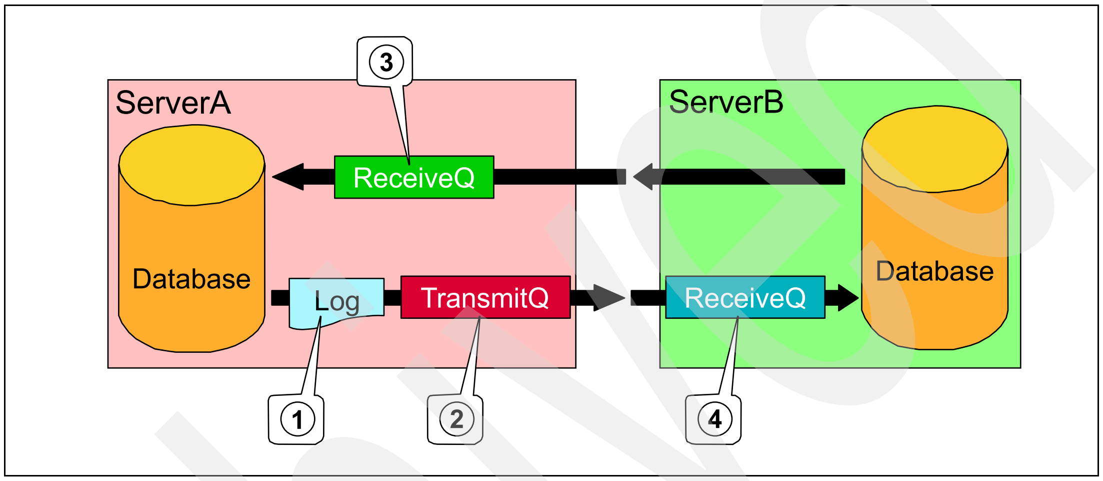


Figure 3-5 Data loss and potential sources of conflicts

Important: Identifying the data loss suffered can be critical for certain types of applications.

When switchback occurs, then the changes from the secondary server that are replicated over to the primary server will pass through the receive queue at the primary server (item 3 in Figure 3-5). These changes may conflict with the changes that may have already occurred at the primary server just prior to failure but were not replicated over to the secondary server (items 1 and 2 in Figure 3-5). Additionally, during switchback, any unpropagated changes on the DB2 log and transmit queue on the primary server (items 1 and 2 in Figure 3-5) may conflict with changes that may have already occurred at the secondary server after failover but prior to switchback.

Therefore, it is possible for conflicts to occur on both the primary server during switchback and the secondary server after failover.

To support the ABC Financials business requirement of failover and switchback, the conflict rule must be set for the primary server to be designated as the loser. Any conflicts will resolve in favor of the updates originating at the secondary server, which would represent the most recent changes to the data.

Attention: The triggering event for failover is assumed to originate external to the Q replication environment. If the switching of the update workload to the secondary server is also automated by external processes, then the likelihood that messages in the receive queue could be drained prior to enabling updating applications is small—conflicts are likely to occur on the secondary server.

At failover, the following events will occur at the secondary server if no specific action is taken:

- The Q Apply program will soon catch up with any messages sent by the primary server, and will have no additional processing to perform until switchback.
- The transmit queue will store all messages sent to the primary server up to the value specified by MAXDEPTH, or until the STORAGE CLASS for the queue is filled.
- Upon the transmit queue reaching its capacity (MAXDEPTH or running out of storage class space), the Q Capture program will act based on the ERROR_ACTION setting in the IBMQREP_SENDQUEUES table:
  - If the ERROR_ACTION is I, all subscriptions using the replication queue map will be deactivated. Switchback from this situation will require a full refresh of all subscriptions. This option is not generally recommended since a transient queue problem will require a reload of all subscriptions.
  - If the ERROR_ACTION is S (default), the Q Capture program will stop. This is the action chosen in our scenario, and will allow a restart without rebuilding any tables.

In order to avoid deactivation of the Q Subscriptions, and subsequent full refresh, the MAXDEPTH and/or storage class size of the transmit queue should be increased to a size capable of accumulating messages for the duration that the primary server is unavailable. The value of MAXDEPTH depends on:

- Amount of file system space available for WebSphere MQ
- Amount of update activity on the system
- Number and size of transactions (including LOBs included in replication)

If the primary server outage is expected to be large and cause the transmit queue to fill up, then Q Capture must be shut down before the transmit queue fills up. Shutting down Q Capture transfers the burden of maintaining transaction information for replication from the WebSphere MQ queues to the DB2 logs. Once Q Capture is shut down, the duration of the primary server outage can last for as long as the DB2 logs are preserved.

Attention: If the amount of time required by Q Capture to catch up to the last committed transaction in the DB2 log exceeds an acceptable switchback time, or the primary server outage lasts for a period greater than the DB2 log retention period, the Q replication configuration may need to be re-initialized, including a full refresh of all tables.

Failovers may be controlled or uncontrolled:

- With controlled failovers, it is possible to ensure that all changes on the primary server are replicated successfully before taking the primary server down. This ensures that there is no “data loss” due to changes in the DB2 log that have not as yet been sent to the WebSphere MQ queue (item 1 in Figure 3-5 on page 35), or messages in the WebSphere MQ queue that have not been transmitted to the secondary server (item 2 in Figure 3-5 on page 35), or messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 3-5 on page 35) when the secondary server is enabled for updates.

  Controlled failover is generally used in scheduled maintenance scenarios.
- Uncontrolled failovers will result in “data loss,” at least until the primary server is restored and switchback processing is completed. The amount of potential “data loss” may be estimated based on Q Capture and the end-to-end latency statistics just prior to failover.

3.5.2 Switchback processing considerations

As mentioned earlier, switchback is the process of restoring the Q replication environment to its normal operating environment after failover processing has occurred. For ABC Financials' bidirectional topology, this involves restoring WTSC53 as the primary server where application updates occur, and WTSC59 as the secondary server containing a read-only replica of the application tables. Switchback should involve minimum data loss.

Important: Identifying the data loss suffered can be critical for certain types of applications such as financials.

As mentioned earlier, when switchback occurs, then the changes from the secondary server that are replicated over to the primary server will pass through the receive queue at the primary server (item 3 in Figure 3-5 on page 35). These changes may conflict with the changes that may have already occurred at the primary server just prior to failure but were not replicated over to the secondary server (items 1 and 2 in Figure 3-5 on page 35). Additionally, during switchback, any unpropagated changes on the DB2 log and transmit queue on the primary server (items 1 and 2 in Figure 3-5) may conflict with changes that may have already occurred at the secondary server after failover but prior to switchback.

This section provides a high-level overview of some of the considerations involved with switchback processing associated with bidirectional replication. Depending upon the particular environment, the process involved in ensuring satisfactory switchback processing can get quite complex.

Note: All switchback operations are controlled, and therefore provide the database administrator the opportunity to carefully plan and execute the most appropriate switchback procedure after a controlled or uncontrolled failover.

Table 3-1 summarizes the considerations in choosing between four possible switchback options.

Table 3-1 Criteria for selecting the appropriate switchback procedure

Switchback option selection criteria	**OPTION A** **=========** **Resync the primary and secondary servers automatically**	**OPTION B** **=========** **Resync the primary with unpropagated changes from the secondary server, and then re-initialize the secondary server with the primary server**	**OPTION C** **=========** **Resync the secondary server with unpropagated changes from the primary server, and then re-initialize the primary server with the secondary server**	**OPTION D** **=========** **Discard the unpropagated changes from the primary server, and then re-initialize the primary server with the secondary server**
Data loss acceptable	No	No	No	Yes
Primary server outage duration	Short	Medium	Large	Large
Potential data loss on the primary server	Small	Medium	High	Do not know
Business value of the data loss on primary server	High	High	High	Low
Workload unavailability duration during switchback	Short	Medium	Large	Large

Switchback option selection criteria	**OPTION A** **=========** **Resync the primary and secondary servers automatically**	**OPTION B** **=========** **Resync the primary with unpropagated changes from the secondary server, and then re-initialize the secondary server with the primary server**	**OPTION C** **=========** **Resync the secondary server with unpropagated changes from the primary server, and then re-initialize the primary server with the secondary server**	**OPTION D** **=========** **Discard the unpropagated changes from the primary server, and then re-initialize the primary server with the secondary server**
Number of exceptions generated	Manageable	Manageable	Manageable	Unmanageable
Ability to abort switchback and stay in failover mode	Poor	Good	Poor	Good

The four options are:

- Option A involves starting Q Apply and warm starting Q Capture on the primary server when it becomes available. It is assumed that Q Apply is running on the secondary server or, if not, that it is started when the primary server becomes available. No data loss is incurred in this scenario.
- Option B involves starting Q Apply on the primary server when it becomes available, but not Q Capture on the primary server right away. It involves re-synchronizing the primary server with unpropagated changes from the secondary server. Once that is completed, the subscriptions are deactivated and reactivated so that the secondary server is re-initialized with the contents of the primary server. No data loss is incurred in this scenario.
- Option C involves warm starting Q Capture on the primary server when it becomes available, but not Q Apply on the primary server right away. It is assumed that Q Apply is running on the secondary server or, if not, that it is started when the primary server becomes available. This option involves re-synchronizing the secondary server with unpropagated changes from the primary server. Once that is completed, the subscriptions are deactivated and reactivated so that the primary server is re-initialized with the contents of the secondary server. No data loss is incurred in this scenario.
- Option D involves discarding all unpropagated changes from the primary server, and re-initializing the primary server with the contents of the secondary server. This is almost like a disaster recovery scenario, except that the primary site becomes available at a later point in time. Data loss is incurred in this scenario.

The criteria mentioned in Table 3-1 are discussed briefly here:

- Data loss acceptable refers to an organization's tolerance for data loss during failover or switchback processing. As mentioned earlier, data loss refers to transactions that are not propagated and applied to the "target" table. (Here "target" does not refer to the location of the table but the object of the replication.) Therefore, changes may have originated on the primary server prior to failover, or originated on the secondary server after failover and prior to switchback. These unpropagated changes are considered to be "data loss," and may be permanent if they cannot be recovered, or temporary if they can be recovered after the failing site is restored.
- Primary server outage duration refers to the duration that the primary server is unavailable. The effort involved in restoring the normal operating environment depends upon the expected duration of the outage of the primary server.

  - If the duration of the primary server outage is short and Q Capture was not shut down on the secondary server, then the bidirectional replication configuration implemented will re-synchronize and resume normal operations simply by restarting the failed server, and starting Q Apply and warm starting Q Capture on the primary server. This is option A.
  - If the duration of the primary server outage is estimated to be so long that Q Capture was shut down to prevent the transmit queue on the secondary server from filling up, then multiple switchback options (B, C, or D) are available, as described earlier.
- Potential data loss on the primary server is the quantification of the data loss if option D[2] is chosen. This is an important consideration in deciding to whether to choose option D.
- Business value of the data loss on primary server attempts to assign a qualitative value to the potential data loss if option D is chosen. This is generally relevant to financial transactions and a very important consideration in deciding to whether to choose option D.
- Workload unavailability duration during switchback refers to the duration that the workload has to be suspended on the secondary server until normal operations can be restored. Each of the different options has a different window of workload unavailability, and may be a deciding factor in the choice of a particular switchback option, for example, option D.
- Number of exceptions generated refers to the number of conflict or SQL error exceptions that may be generated and will have to be resolved. The number of exceptions depends upon the potential data loss on the primary server after failover, and the duration of the primary server outage. The greater the potential data loss, and the larger the primary server outage, the greater the potential number of exceptions generated.

Ability to abort switchback and stay in failover mode refers to the ability of the database administrator to abandon switchback processing in case of problems, and stay with processing on the secondary server. As long as unpropagated changes from the primary server are not replicated to the secondary server, the database administrator could abandon switchback processing and continue with the workload on the secondary server. This is possible with options B and D, but not with options A and C.

Important: Options B, C, and D involve cold starting Q Capture, which is generally not recommended. However, we recognize that you may experience extenuating circumstances that may warrant your taking one of these approaches, and have therefore documented the steps involved. We expect option A to be the norm in switchback operations.

Figure 3-6 summarizes the failover and switchback options discussed earlier.

[2] Options A, B, and C do not incur data loss.

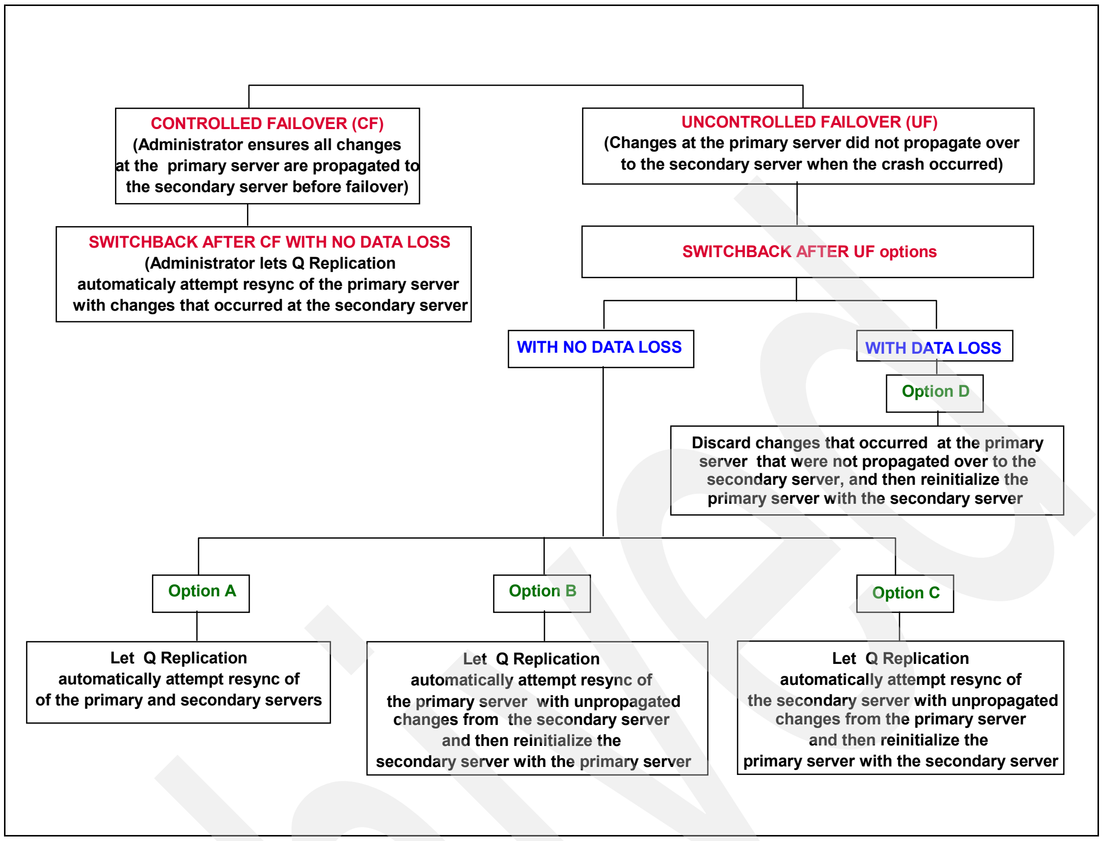

CONTROLLED FAILOVER (CF)
(Administrator ensures all changes at the primary server are propagated to the secondary server before failover)
UNCONTROLLED FAILOVER (UF)
(Changes at the primary server did not propagate over to the secondary server when the crash occurred)
SWITCHBACK AFTER CF WITH NO DATA LOSS
(Administrator lets Q Replication automaticaly attempt resync of the primary server with changes that occurred at the secondary server
SWITCHBACK AFTER UF options
WITH NO DATA LOSS
WITH DATA LOSS
Option D
Discard changes that occurred at the primary server that were not propagated over to the secondary server, and then reinitialize the primary server with the secondary server
Option A
Let Q Replication automatically attempt resync of of the primary and secondary servers
Option B
Let Q Replication automatically attempt resync of the primary server with unpropagated changes from the secondary server and then reinitialize the secondary server with the primary server
Option C
Let Q Replication automatically attempt resync of the secondary server with unpropagated changes from the primary server and then reinitialize the primary server with the secondary server

Figure 3-6 Bidirectional replication high availability failover and switchback options

3.6 Controlled failover and switchback

Controlled failover is generally used in scheduled maintenance and migration scenarios, where an organization needs to upgrade the hardware/software of the primary server while providing almost uninterrupted business application service on a secondary server in the interim. The duration of such a scheduled outage of the primary server may be short or long depending upon the extent of the upgrades planned.

Given the planned nature of the failover, procedures can be implemented to eliminate data loss and potential conflicts on the secondary server prior to failover.

The following subsections describe the recommended procedure for a controlled failover and switchback from the controlled failover.

3.6.1 Controlled failover (CF)

The recommended controlled failover procedure is shown in Figure 3-7 and assumes the following:

- Bidirectional replication is up and running on the primary and secondary server. All subscriptions are active and Q Capture and Q Apply are running on both the primary and secondary servers.
- Primary server is operating in active mode. It has an update workload running, while the secondary server is read-only.

- Replication is performing well with the end-to-end latency rate is in seconds. It is being monitored and the metrics are acceptable.
- Controlled failover is performed during a maintenance period when the throughput is low.
- There are no conflict exceptions resulting from this failover.

Attention: The guiding principle of this failover procedure is to minimize the potential for exceptions and contention on the secondary server.

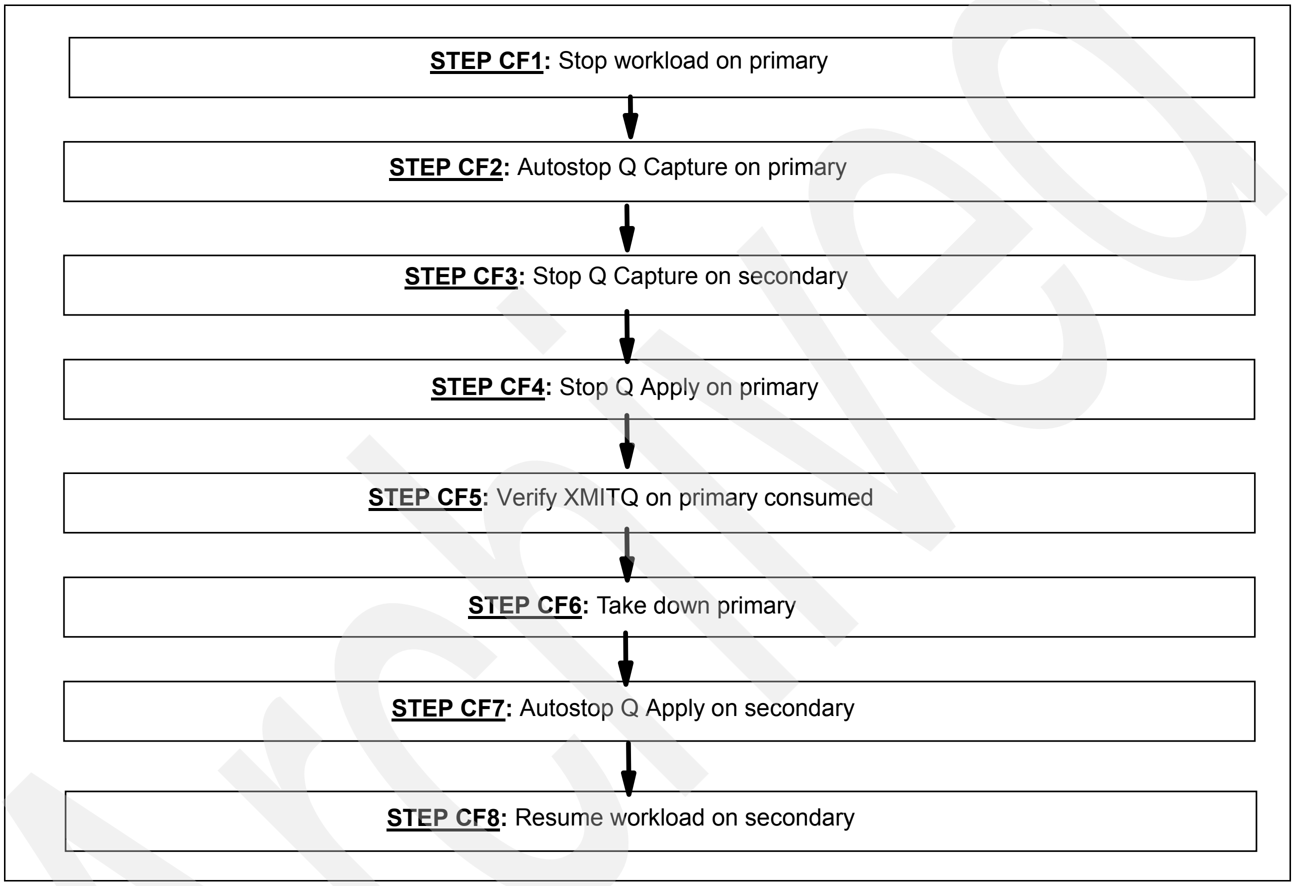

STEP CF1: Stop workload on primary
STEP CF2: Autostop Q Capture on primary
STEP CF3: Stop Q Capture on secondary
STEP CF4: Stop Q Apply on primary
STEP CF5: Verify XMITQ on primary consumed
STEP CF6: Take down primary
STEP CF7: Autostop Q Apply on secondary
STEP CF8: Resume workload on secondary

Figure 3-7 Overview of controlled failover (CF) steps

Each of the steps shown in Figure 3-7 is described briefly in the following sections.

STEP CF1: Stop workload on primary

The first step is to stop the application workload directed to this server without redirecting it to the secondary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP CF2: Autostop Q Capture on primary

In this step, the Q Capture program on the primary server is directed to shut down after it has completed processing all the changes that have been committed while the workload was running, and written them to the send queue.

Example 3-2 shows the command that can be used to achieve this function. To determine whether the Q Capture program has stopped, you should monitor the contents of the Q Capture log, which shows the program as having stopped (as shown in Example 3-3).

Example 3-2 Autostop Q Capture on the primary server SC53 - modify command

```
F QCAPITSO, CHGPARMS AUTOSTOP=Y
ASN0523I  "Q Capture" : "ITSO" : "Initial" : The CHGPARMS command
response: "AUTOSTOP" has been set to "Y".
ASN0573I  "Q Capture" : "ITSO" : "Initial" : The program was stopped.
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QCAPITSO          QCAP        00   3419   3.16    .00    9.9  2188K
 0      0      0      0     0
IEF404I QCAPITSO - ENDED - TIME=19.31.31 - ASID=0032 - SC53
-QCAPITSO ENDED.  NAME-                     TOTAL CPU TIME=  3.16
TOTAL ELAPSED TIME=   9.9
$HASP395 QCAPITSO ENDED
```

Example 3-3 Autostop Q Capture log contents on the primary server SC53

```
....................
2006-02-23-19.31.15.636044 <handleCHGPARMS> ASN0523I  "Q Capture" : "ITSO" : "Initial" : The CHGPARMS
command response: "AUTOSTOP" has been set to "Y".
2006-02-23-19.31.22.349454 <asnqwk> ASN7020I  "Q Capture" : "ITSO" : "WorkerThread" : The program reached
the end of the active log and terminated because the AUTOSTOP option is specified.
2006-02-23-19.31.22.436919 <asnqwk> ASN7109I  "Q Capture" : "ITSO" : "WorkerThread" :  At program
termination, the highest log sequence number of a successfully processed transaction is
"BE68:F905:6E69:0001" and the lowest log sequence number of a transaction still to be committed is
"BE68:FC36:1955:0001".
...............
2006-02-23-19.31.30.668891 <asnqcap::main> ASN0573I  "Q Capture" : "ITSO" : "Initial" : The program was
stopped.
```

STEP CF3: Stop Q Capture on secondary

In this step, an immediate stop is issued for the Q Capture program on the secondary server. Since the secondary server is read-only, there should not be any changes recorded on the DB2 log.

Example 3-4 shows the command that can be used to stop Q Capture on the secondary server. To determine whether the Q Capture program has stopped, you should monitor the contents of the Q Capture log, which shows the program as having stopped (as shown in Example 3-5).

Example 3-4 Stop Q Capture on the secondary server SC59 - modify command

```
F QCAPITSO, STOP
ASN0522I  "Q Capture" : "ITSO" : "Initial" : The program received the
"STOP" command.
ASN0573I  "Q Capture" : "ITSO" : "Initial" : The program was stopped.
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QCAPITSO          QCAP        00  11029   7.17    .00   21.9  1004M
 0      0      0      0     0
-QCAPITSO ENDED.  NAME-HUEYKIM              TOTAL CPU TIME=  7.17
```

```
TOTAL ELAPSED TIME=  21.9
$HASP395 QCAPITSO ENDED
$HASP309 INIT 1    INACTIVE ******** C=ABCDEFG12345
SE '19.48.58 JOB13231 $HASP165 QCAPITSO ENDED AT WTSC59  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

Example 3-5 Stop Q Capture log contents on the secondary server SC59

```
....................
2006-02-23-19.48.53.339500 <handleZOsEvent> ASN0522I  "Q Capture" : "ITSO" : "Initial" : The program
received the "STOP" command.
2006-02-23-19.48.53.509300 <asnqwk> ASN7109I  "Q Capture" : "ITSO" : "WorkerThread" :  At program
termination, the highest log sequence number of a successfully processed transaction is
"0000:0BFD:AE2E:0000" and the lowest log sequence number of a transaction still to be committed is
"0000:0C48:B0C4:0000".
................
2006-02-23-19.48.58.350348 <asnqcap::main> ASN0573I  "Q Capture" : "ITSO" : "Initial" : The program was
stopped.
```

STEP CF4: Stop Q Apply on primary

In this step, an immediate stop is issued for the Q Apply program on the primary server.

Example 3-6 shows the command that can be used to stop Q Apply on the primary server. To determine whether the Q Apply program has stopped, you should monitor the contents of the Q Apply log, which shows the program as having stopped (as shown in Example 3-7).

Example 3-6 Stop Q Apply on the primary server SC53 - modify command

```
F QAPPITSO, STOP
ASN0522I  "Q Apply" : "ITSO" : "Initial" : The program received the
"STOP" command.
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QAPPITSO          QAPP        00  15964    .05    .00   31.2  47442
 0      0      0      0     0
IEF404I QAPPITSO - ENDED - TIME=19.52.39 - ASID=0031 - SC53
-QAPPITSO ENDED.  NAME-                     TOTAL CPU TIME=   .05
TOTAL ELAPSED TIME=  31.2
$HASP395 QAPPITSO ENDED
SE '19.52.39 JOB25868 $HASP165 QAPPITSO ENDED AT WTSCPLX1  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

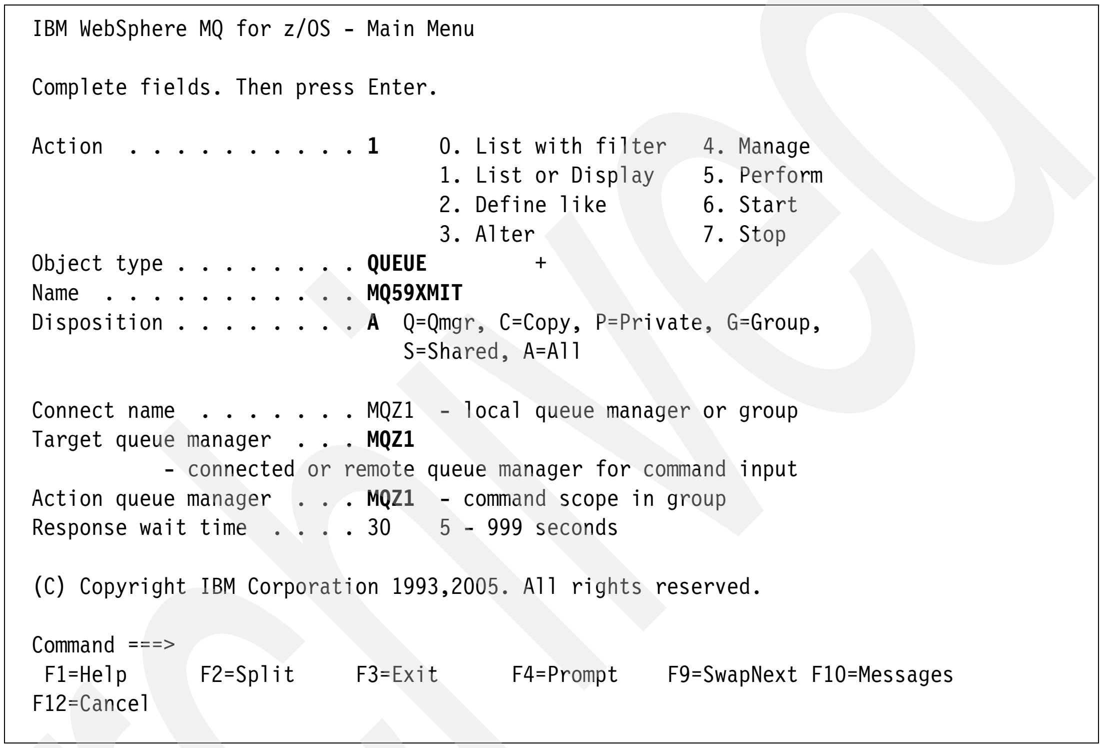
Example 3-7 Stop Q Apply log contents on the primary server SC53

```
....................
2006-02-23-19.52.32.585159 <handleZOsEvent> ASN0522I  "Q Apply" : "ITSO" : "Initial" : The program received
the "STOP" command.
.....................
2006-02-23-19.52.35.890715 <browser::~browser> ASN8999D  "Q Apply" : "ITSO" : "BR00000" : browser for queue
'QREP.POKB.TO.POKA.RECVQ' found max msgID '5152455043FE4EC5000000000000000000000000000000023' after cleaning
IBMQREP_DONEMSG table on shutdown.
2006-02-23-19.52.35.903020 <brwzMain> ASN8999D  "Q Apply" : "ITSO" : "BR00000" : browser 'BR00000' for
queue 'QREP.POKB.TO.POKA.RECVQ' terminating with code 0
...........................
2006-02-23-19.52.37.601377 <asnqapp::main> ASN0573I  "Q Apply" : "ITSO" : "Initial" : The program was
stopped.
```

STEP CF5: Verify XMITQ on primary consumed

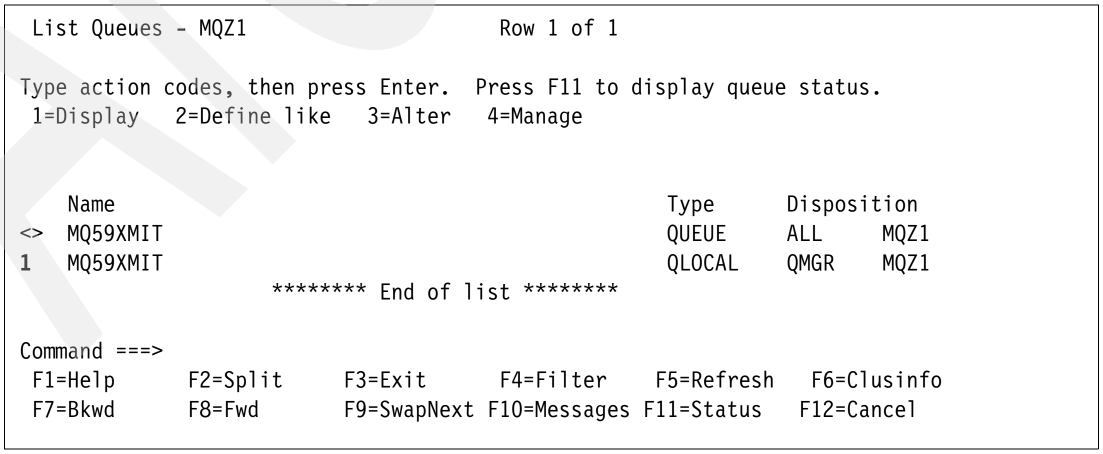
At this point, only the Q Apply on the secondary server is running and consuming messages received from the primary server.

Before the primary server can be taken offline for maintenance, you must ensure that all the messages in the transmit queue on the primary server have been propagated over to the secondary server. Figure 3-8 through Figure 3-10 show the WebSphere MQ panels to navigate to determine that the transmit queue is empty. The value of the Current queue depth field in Figure 3-10 is zero indicating that the transmit queue MQ59XMIT on the primary server has been fully consumed.



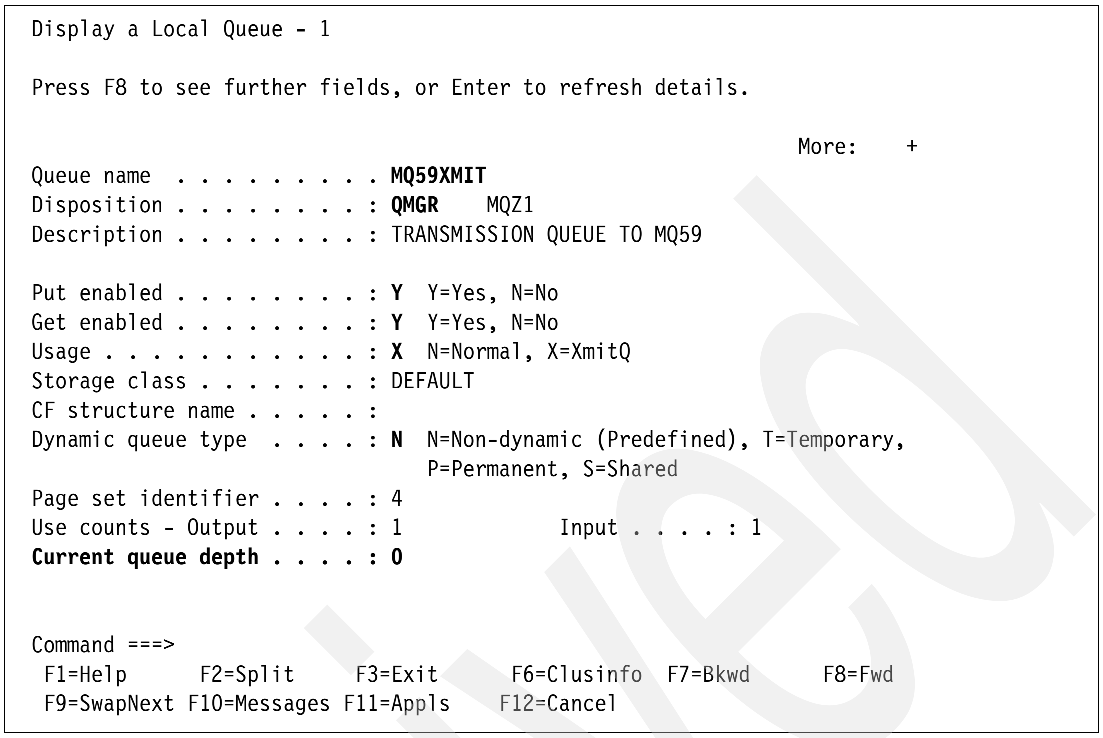
```
 IBM WebSphere MQ for z/OS - Main Menu

 Complete fields. Then press Enter.

 Action  . . . . . . . . . . 1     0. List with filter   4. Manage
                                   1. List or Display    5. Perform
                                   2. Define like        6. Start
                                   3. Alter              7. Stop
 Object type . . . . . . . . QUEUE        +
 Name  . . . . . . . . . . . MQ59XMIT
 Disposition . . . . . . . . A  Q=Qmgr, C=Copy, P=Private, G=Group,
                                S=Shared, A=All

 Connect name  . . . . . . . MQZ1  - local queue manager or group
 Target queue manager  . . . MQZ1
           - connected or remote queue manager for command input
 Action queue manager  . . . MQZ1  - command scope in group
 Response wait time  . . . . 30    5 - 999 seconds

 (C) Copyright IBM Corporation 1993,2005. All rights reserved.

 Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
 F12=Cancel
```

Figure 3-8 Verify transmit queue on the primary server SC53 is fully consumed 1/3



```
 List Queues - MQZ1                    Row 1 of 1

Type action codes, then press Enter.  Press F11 to display queue status.
 1=Display   2=Define like   3=Alter   4=Manage


    Name                                         Type      Disposition
<>  MQ59XMIT                                     QUEUE     ALL     MQZ1
1   MQ59XMIT                                     QLOCAL    QMGR    MQZ1
                     ******** End of list ********

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F6=Clusinfo
 F7=Bkwd      F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 3-9 Verify transmit queue on the primary server SC53 is fully consumed 2/3



```
Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

                                                                More:    +
Queue name  . . . . . . . . . MQ59XMIT
Disposition . . . . . . . . : QMGR    MQZ1
Description . . . . . . . . : TRANSMISSION QUEUE TO MQ59

Put enabled . . . . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . . . . : Y  Y=Yes, N=No
Usage . . . . . . . . . . . : X  N=Normal, X=XmitQ
Storage class . . . . . . . : DEFAULT
CF structure name . . . . . :
Dynamic queue type  . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                 P=Permanent, S=Shared
Page set identifier . . . . : 4
Use counts - Output . . . . : 1            Input . . . . : 1
Current queue depth . . . . : 0


Command ===>
 F1=Help      F2=Split     F3=Exit      F6=Clusinfo  F7=Bkwd      F8=Fwd
 F9=SwapNext F10=Messages F11=Appls    F12=Cancel
```

Figure 3-10 Verify transmit queue on the primary server SC53 is fully consumed 3/3

STEP CF6: Take down primary

The primary server can now be taken offline.

STEP CF7: Autostop Q Apply on secondary

In this step, the Q Apply program on the secondary server is directed to shut down after it has completed processing all the changes propagated from the primary server.

Example 3-8 shows the command that can be used to achieve this function. To determine whether the Q Apply program has stopped, you should monitor the contents of the Q Apply log, which shows the program as having stopped, as shown in Example 3-9.

Example 3-8 Autostop Q Apply on the secondary server SC59 - modify command

```
F QAPPITSO, CHGPARMS AUTOSTOP=Y
ASN0523I  "Q Apply" : "ITSO" : "Initial" : The CHGPARMS command
response: "AUTOSTOP" has been set to "Y".
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QAPPITSO          QAPP        00   9265    .03    .00   13.0  4908K
 0      0      0      0     0
-QAPPITSO ENDED.  NAME-HUEYKIM             TOTAL CPU TIME=   .03
TOTAL ELAPSED TIME=  13.0
$HASP395 QAPPITSO ENDED
$HASP309 INIT 2    INACTIVE ******** C=AB
SE '20.00.11 JOB13234 $HASP165 QAPPITSO ENDED AT WTSC59  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

Example 3-9 Autostop Q Apply log contents on the secondary server SC59

```
....................
2006-02-23-20.00.02.363998 <handleCHGPARMS> ASN0523I  "Q Apply" : "ITSO" : "Initial" : The
CHGPARMS command response: "AUTOSTOP" has been set to "Y".
2006-02-23-20.00.03.549406 <brwzMain> ASN8999D  "Q Apply" : "ITSO" : "BR00000" : Browser
for queue 'QREP.POKA.TO.POKB.RECVQ' will be suspended because AUTOSTOP option was
specified.
2006-02-23-20.00.05.549829 <brwzMain> ASN7590I  "Q Apply" : "ITSO" : "BR00000" : The Q
Apply program stopped reading from the queue "QREP.POKA.TO.POKB.RECVQ" for replication
queue map "QMAP_POKA_TO_POKB". Reason code: "0".
.........................
2006-02-23-20.00.07.554676 <browser::~browser> ASN8999D  "Q Apply" : "ITSO" : "BR00000" :
browser for queue 'QREP.POKA.TO.POKB.RECVQ' found max msgID
'5152455043FE4EC300000000000000000000000000000052' after cleaning IBMQREP_DONEMSG table on
shutdown.
............................
2006-02-23-20.00.11.380407 <asnqapp::main> ASN0573I  "Q Apply" : "ITSO" : "Initial" : The
program was stopped.
```

STEP CF8: Resume workload on secondary

The application workload can now be started on the secondary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

3.6.2 Switchback after CF with no data loss

Given the controlled nature of the failover, the procedure for switching back to the primary server and establishing normal operations is somewhat simpler than in the case of switchback after an uncontrolled failover.

The recommended switchback procedure after a successful controlled failover is shown in Figure 3-11 and assumes the following:

- The primary server is up and running DB2.
- WebSphere MQ is up and running on both the primary and secondary servers.
- The primary server is not running any workload—read-only or update.
- Q Capture and Q Apply are not running on either the primary or secondary servers.
- All the subscriptions are in an active state.
- The application workload is running on the secondary server causing changes to be written to the DB2 log.
- Switchback is performed during a maintenance period when the throughput is low.
- There are no conflict exceptions resulting from this switchback.

Attention: The guiding principle of this switchback procedure is to:

- Minimize the potential for exceptions and contention on the primary server.
- Narrow the window of application workload unavailability by stopping the workload as late as possible and restarting it as early as possible.

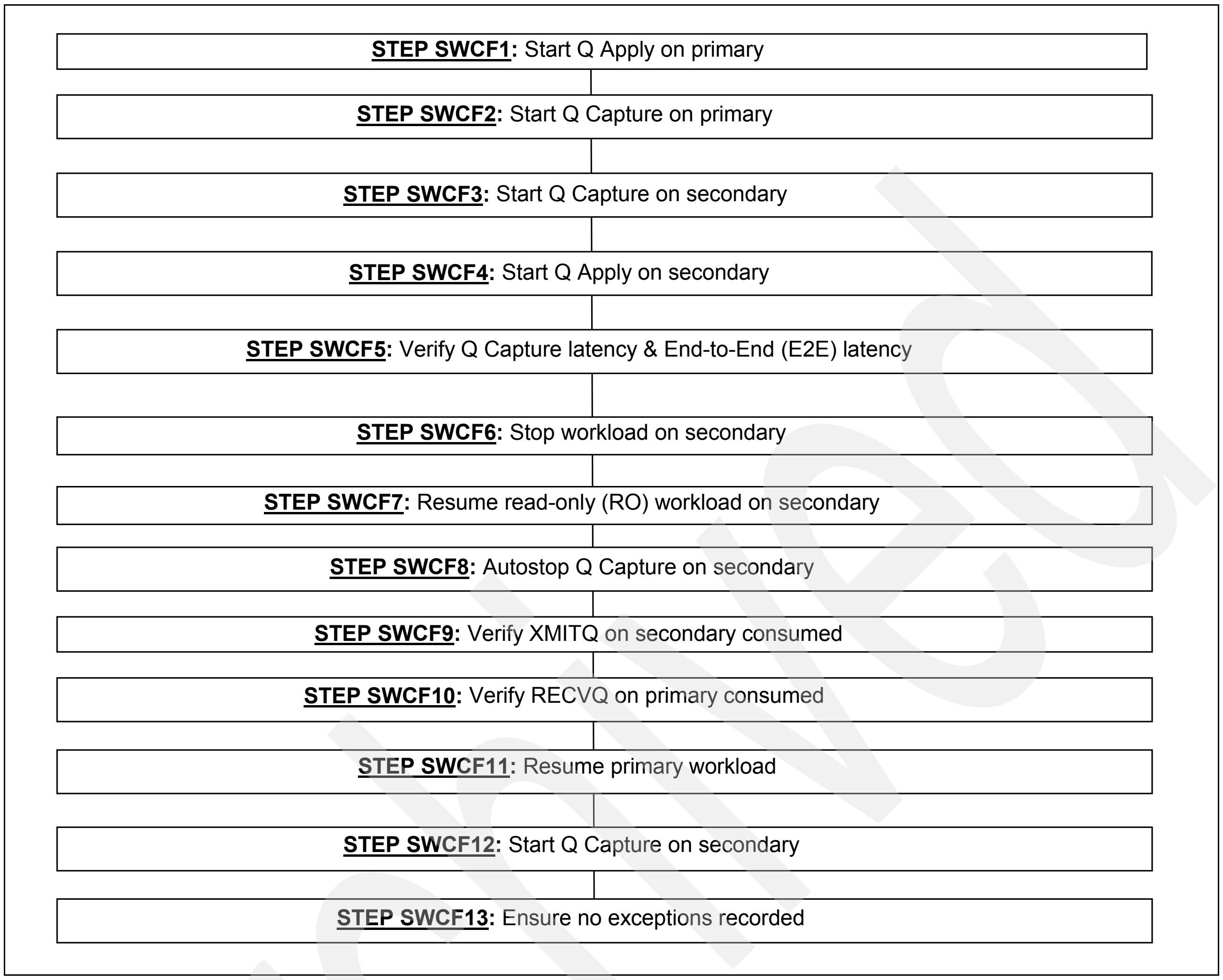

STEP SWCF1: Start Q Apply on primary
STEP SWCF2: Start Q Capture on primary
STEP SWCF3: Start Q Capture on secondary
STEP SWCF4: Start Q Apply on secondary
STEP SWCF5: Verify Q Capture latency & End-to-End (E2E) latency
STEP SWCF6: Stop workload on secondary
STEP SWCF7: Resume read-only (RO) workload on secondary
STEP SWCF8: Autostop Q Capture on secondary
STEP SWCF9: Verify XMITQ on secondary consumed
STEP SWCF10: Verify RECVQ on primary consumed
STEP SWCF11: Resume primary workload
STEP SWCF12: Start Q Capture on secondary
STEP SWCF13: Ensure no exceptions recorded

Figure 3-11 Overview of switchback after controlled failover (CF) steps

Each of the steps shown in Figure 3-11 is described briefly in the following sections.

STEP SWCF1: Start Q Apply on primary

Start Q Apply on the primary server by submitting the job shown in Example 3-10, the output of which indicates that the Q Apply program has started processing the receive queue QREP.POKB.TO.POKA.RECVQ, as shown in Example 3-11. You can monitor the contents of the Q Apply log, which shows the Q Apply program initializing successfully and the various agents ready to process the receive queue, as shown in Example 3-12.

The status of the Q Apply program may be viewed by issuing the **asnqacmd** command, which displays the state of the various threads associated with the Q Apply process, as shown in Example 3-13.

Example 3-10 JCL to start Q Apply on the primary server SC53

```
//QAPPITSO JOB NOTIFY=&SYSUID,                                        JOB17779
//         MSGCLASS=H,MSGLEVEL=(1,0),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//QAPP EXEC PGM=ASNQAPP,
//  PARM='/APPLY_SERVER=D8G1 APPLY_PATH=//''HUEYKIM
//             APPLY_SCHEMA=ITSO'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//         DD DSN=MQ600.SCSQANLE,DISP=SHR
//         DD DSN=MQ600.SCSQLOAD,DISP=SHR
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

Example 3-11   Start Q Apply on the primary server SC53 - job submission

```
$HASP100 QAPPITSO ON INTRDR                            FROM TSU25559
HUEYKIM
IRR010I  USERID HUEYKIM  IS ASSIGNED TO THIS JOB.
ICH70001I HUEYKIM  LAST ACCESS AT 19:38:04 ON THURSDAY, FEBRUARY 23,
2006
$HASP373 QAPPITSO STARTED - INIT A    - CLASS A - SYS SC53
IEF403I QAPPITSO - STARTED - TIME=20.11.58 - ASID=0031 - SC53
ASN0559W  "Q Apply" : "N/A" : "Initial" : The job was started 294
with a CPU time limit of "25750" seconds. The program will
terminate when the time limit expires.
$HASP100 BPXAS    ON STCINRDR
$HASP373 BPXAS    STARTED
IEF403I BPXAS - STARTED - TIME=20.12.00 - ASID=0071 - SC53
$HASP100 BPXAS    ON STCINRDR
$HASP373 BPXAS    STARTED
IEF403I BPXAS - STARTED - TIME=20.12.01 - ASID=006E - SC53
2006-02-23-20.12.02.586388 ASN8999D  "Q Apply" : "N/A" : 301
"Initial" : thread name = Initial, tid = 0x12367c6000000000, tcb
= 006D2C48
ASN7526I  "Q Apply" : "ITSO" : "BR00000" : The Q Apply program 302
has started processing the receive queue"QREP.POKB.TO.POKA.RECVQ"
for replication queue map "QMAP_POKB_TO_POKA".
```

Example 3-12   Start Q Apply log contents on the primary server SC53

```
.......................
2006-02-23-20.12.05.006715 <asnqapp::main> ASN0572I  "Q Apply" : "ITSO" : "Initial" : The
program initialized successfully.
.......................
2006-02-23-20.12.05.105120 <browser::browser> ASN8999D  "Q Apply" : "ITSO" : "BR00000" :
browser for queue 'QREP.POKB.TO.POKA.RECVQ' found max msgID
'5152455043FE4EC5000000000000000000000000000000023' after cleaning IBMQREP_DONEMSG table on
startup.
..........................
2006-02-23-20.12.05.148430 <brwzMain> ASN7526I  "Q Apply" : "ITSO" : "BR00000" : The Q
Apply program has started processing the receive queue"QREP.POKB.TO.POKA.RECVQ" for
replication queue map "QMAP_POKB_TO_POKA".
.................................
2006-02-23-20.12.05.248498 <appAgntMain> ASN8999D  "Q Apply" : "ITSO" : "BR00000AG013" :
agent 013 started for queue "QREP.POKB.TO.POKA.RECVQ"
```

Example 3-13   ASNQACMD command output status

```
HUEYKIM @ SC53:/SC53/tmp>asnqacmd apply_server=d8g1 apply_schema=itso status
2006-02-23-20.55.11.993961 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x123d9af000000000, tcb = 006FD098
```

```
2006-02-23-20.55.14.047936 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "HoldLThread" thread is in the "is waiting" state.
2006-02-23-20.55.14.050280 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
2006-02-23-20.55.14.051541 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "MonitorThread" thread is in the "is resting" state.
2006-02-23-20.55.14.053730 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "BR00000" thread is in the "is doing work" state.
```

STEP SWCF2: Start Q Capture on primary

Start Q Capture on the primary server by submitting the job shown in Example 3-14, the output of which indicates that the Q Capture program initialized successfully, as shown in Example 3-15. You can monitor the contents of the Q Capture log, which shows the Q Capture program initializing successfully, as shown in Example 3-16.

The status of the Q Capture program may be viewed by issuing the **asnqccmd** command, which displays the state of the various threads associated with the Q Capture process, as shown in Example 3-17.

Example 3-14   JCL to start Q Capture on the primary server SC53

```
//QCAPITSO JOB NOTIFY=&SYSUID,                                      JOB17436
//         MSGCLASS=H,MSGLEVEL=(1,0),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//QCAP     EXEC PGM=ASNQCAP,
// PARM='/CAPTURE_SERVER=D8G1 capture_schema=ITSO startmode=warmsi
//             CAPTURE_PATH=//''HUEYKIM'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//         DD DSN=MQ600.SCSQANLE,DISP=SHR
//         DD DSN=MQ600.SCSQLOAD,DISP=SHR
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,70)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

Example 3-15   Start Q Capture on the primary server SC53 - job submission output

```
$HASP100 QCAPITSO ON INTRDR                            FROM TSU25559
HUEYKIM
IRR010I  USERID HUEYKIM  IS ASSIGNED TO THIS JOB.
ICH70001I HUEYKIM  LAST ACCESS AT 20:11:58 ON THURSDAY, FEBRUARY 23,
2006
$HASP373 QCAPITSO STARTED - INIT A    - CLASS A - SYS SC53
IEF403I QCAPITSO - STARTED - TIME=20.45.56 - ASID=0032 - SC53
$HASP100 BPXAS    ON STCINRDR
$HASP373 BPXAS    STARTED
IEF403I BPXAS - STARTED - TIME=20.45.58 - ASID=0071 - SC53
$HASP100 BPXAS    ON STCINRDR
$HASP373 BPXAS    STARTED
IEF403I BPXAS - STARTED - TIME=20.45.58 - ASID=006E - SC53
IEA631I  OPERATOR HUEYKIM  NOW INACTIVE, SYSTEM=SC53    , LU=SC38TCCB
ASN7000I  "Q Capture" : "ITSO" : "WorkerThread" : "4" 392
```

```
subscriptions are active. "0" subscriptions are inactive. "0"
subscriptions that were new and were successfully activated. "0"
subscriptions that were new could not be activated and are now
inactive.
ASN0572I  "Q Capture" : "ITSO" : "WorkerThread" : The program
initialized successfully.
```

Example 3-16   Start Q Capture log contents on the primary server SC53

```
.......................
2006-02-23-20.46.02.295672 <asnParmClass::printParms> ASN0529I  "Q Capture" : "ITSO" :
"Initial" : The value of "STARTMODE" was set to "WARMSI" at startup by the following
method: "COMMANDLINE".
.........................
2006-02-23-20.46.02.469293 <subMgr::processSubscriptions> ASN7000I  "Q Capture" : "ITSO" :
"WorkerThread" : "4" subscriptions are active. "0" subscriptions are inactive. "0"
subscriptions that were new and were successfully activated. "0" subscriptions that were
new could not be activated and are now inactive.
2006-02-23-20.46.02.482242 <thrdPrologue> ASN8999D  "Q Capture" : "ITSO" : "txrdThread" :
thread name = txrdThread, tid = 0x123d4fa000000005, tcb = 006D2348
2006-02-23-20.46.02.491333 <waitForLogrdInit> ASN7108I  "Q Capture" : "ITSO" :
"WorkerThread" :  At program initialization, the highest log sequence number of a
successfully processed transaction is "BE68:F905:6E69:0001" and the lowest log sequence
number of a transaction still to be committed is "BE68:FC36:1955:0001".
2006-02-23-20.46.02.491563 <asnqwk> ASN0572I  "Q Capture" : "ITSO" : "WorkerThread" : The
program initialized successfully.
```

Example 3-17   ASNQCCMD command output status

```
HUEYKIM @ SC53:/u/hueykim>asnqccmd capture_server=d8g1 capture_schema=itso status
2006-02-23-20.57.04.581784 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x123d9af000000000, tcb = 006FD098
2006-02-23-20.57.06.609763 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "HoldLThread" thread is in the "is waiting" state.
2006-02-23-20.57.06.611686 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
2006-02-23-20.57.06.613729 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "PruneThread" thread is in the "is resting" state.
2006-02-23-20.57.06.616158 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "WorkerThread" thread is in the "is doing work" state.
2006-02-23-20.57.06.629903 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "txrdThread" thread is in the "is doing work" state.
```

STEP SWCF3: Start Q Capture on secondary

Start Q Capture on the secondary server by submitting the job shown in Example 3-18, the output of which indicates that the Q Capture program initialized successfully similar to that shown in Example 3-15 on page 49. You can also monitor the contents of the Q Capture log which shows the Q Capture program initializing successfully, similar to that shown in Example 3-16 on page 50.

The status of the Q Capture program may also be viewed by issuing the **asnqccmd** command, which displays the state of the various threads associated with the Q Capture process, similar to that shown in Example 3-17 on page 50.

Example 3-18   JCL to start Q Capture on the secondary server SC59

```
//QCAPITSO JOB (POK,999),HUEYKIM,MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M,TIME=NOLIMIT
//QCAP     EXEC PGM=ASNQCAP,
// PARM='/CAPTURE_SERVER=DB8A capture_schema=ITSO startmode=warmsi
//             CAPTURE_PATH=//''HUEYKIM'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8A8.SDSNLOAD,DISP=SHR
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,70)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
```

STEP SWCF4: Start Q Apply on secondary

Start Q Apply on the secondary server by submitting the job shown in Example 3-19, the output of which indicates that the Q Apply program has started processing the receive queue QREP.POKA.TO.POKB.RECVQ, similar to that shown in Example 3-11 on page 48. You can monitor the contents of the Q Apply log, which shows the Q Apply program initializing successfully and the various agents ready to process the receive queue, similar to that shown in Example 3-12 on page 48.

The status of the Q Apply program may be viewed by issuing the **asnqacmd** command, which displays the state of the various threads associated with the Q Apply process similar to that shown in Example 3-13.

Example 3-19 JCL to start Q Apply on the secondary server SC59

```
//QAPPITSO JOB (POK,999),HUEYKIM,MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M,TIME=NOLIMIT
//QAPP EXEC PGM=ASNQAPP,
//  PARM='/APPLY_SERVER=DB8A APPLY_PATH=//''HUEYKIM
//             APPLY_SCHEMA=ITSO'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8A8.SDSNLOAD,DISP=SHR
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

STEP SWCF5: Verify Q Capture latency and E2E latency

In order to minimize the window during which the business application is not available to end users, it is desirable to shut it down on the secondary server after most of the unpropagated changes have been replicated over to the primary server.

The Q Capture latency and end-to-end latency may be determined using any of the following three methods:

- **asnqacmd** show details command as shown in Example 3-20

  Note: All the **asnqxcmd** commands were executed for z/OS with a UNIX ID.

- Replication Center GUI, as shown in Figure 3-12 on page 53 through Figure 3-15 on page 55

- Live Monitor tool invoked via the **QApplyMonitor** command on a Windows workstation, as shown in Example 3-21 on page 55, and its corresponding graphical output, as shown in Figure 3-16 on page 56

When the Q Capture latency and end-to-end latency is in a few seconds at most, you can conclude that most of the unpropagated changes have been successfully replicated over to the primary server.

Note: Latency statistics are updated at intervals specified by the MONITOR_INTERVAL; therefore, at least one monitor interval must elapse before latency statistics are available.

Example 3-20 Showing Q Capture latency and end-to-end latency using ASNQACMD command details output

```
HUEYKIM @ SC53:/u/hueykim>asnqacmd apply_server=d8g1 apply_schema=itso status show details
2006-02-23-21.15.50.198459 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x123d9af000000000, tcb = 006FD098
Q Apply program status
  Server name                                  (SERVER) = D8G1
  Schema name                                  (SCHEMA) = ITSO
  Program status                               (STATUS) = Up
  Time since program started                  (UP_TIME) =   0d  1h  3m 51s
  Log file location                           (LOGFILE) = //'HUEYKIM.D8G1.ITSO.QAPP.log'
  Number of active Q subscriptions        (ACTIVE_QSUBS) = 4
  Time period used to calculate average (INTERVAL_LENGTH) =  0h  3m 48.422s

  Receive queue : QREP.POKB.TO.POKA.RECVQ
      Number of active Q subscriptions             (ACTIVE_QSUBS) = 4
      All transactions applied as of (time)        (OLDEST_TRANS) =
2006-02-23-21.15.26.000000
      All transactions applied as of (LSN) (ALL_APPLIED_AS_OF_LSN) = 0000:0000:0000:0000
      Oldest in-progress transaction          (OLDEST_INFLT_TRANS) =
1900-01-01-00.00.00.000000
      Average end-to-end latency                 (END2END LATENCY) =  0h  0m  0.0s
      Average Q Capture latency                  (CAPTURE_LATENCY) =  0h  0m  0.0s
      Average WSMQ latency                              (QLATENCY) =  0h  0m  0.0s
      Average Q Apply latency                      (APPLY_LATENCY) =  0h  0m  0.0s
      Current memory                             (CURRENT_MEMORY ) = 0 MB
      Current queue depth                                 (QDEPTH) = 0
```

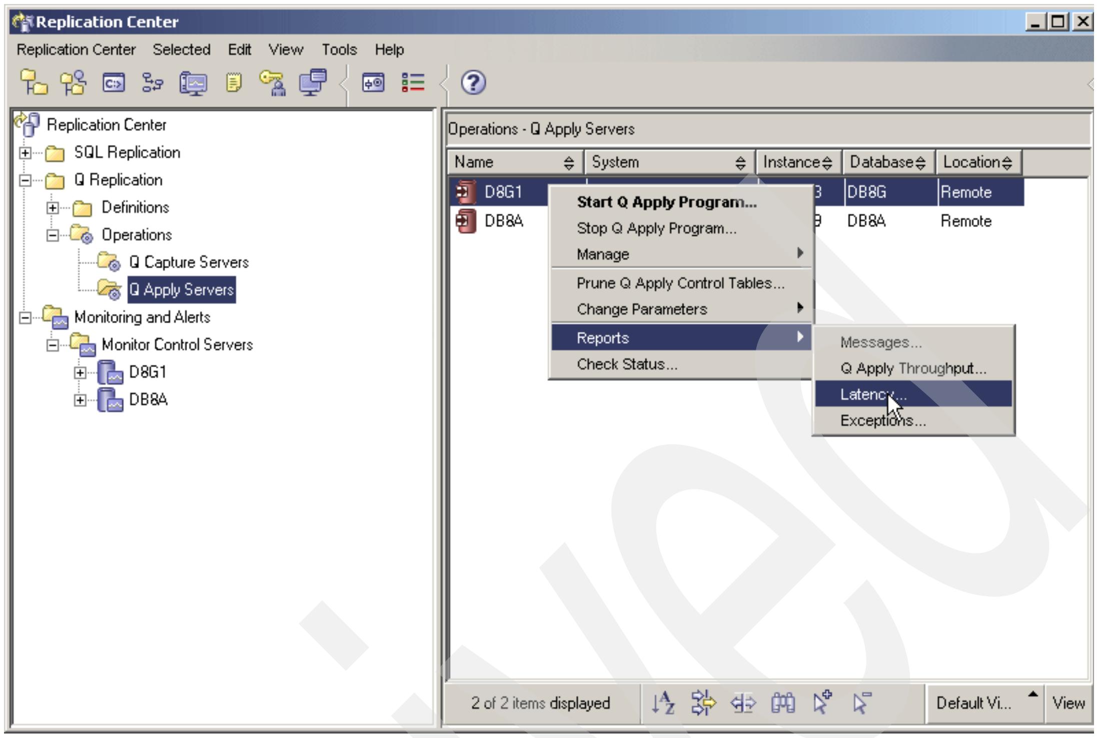


Figure 3-12 Q Capture latency and end-to-end latency using Replication Center 1/4

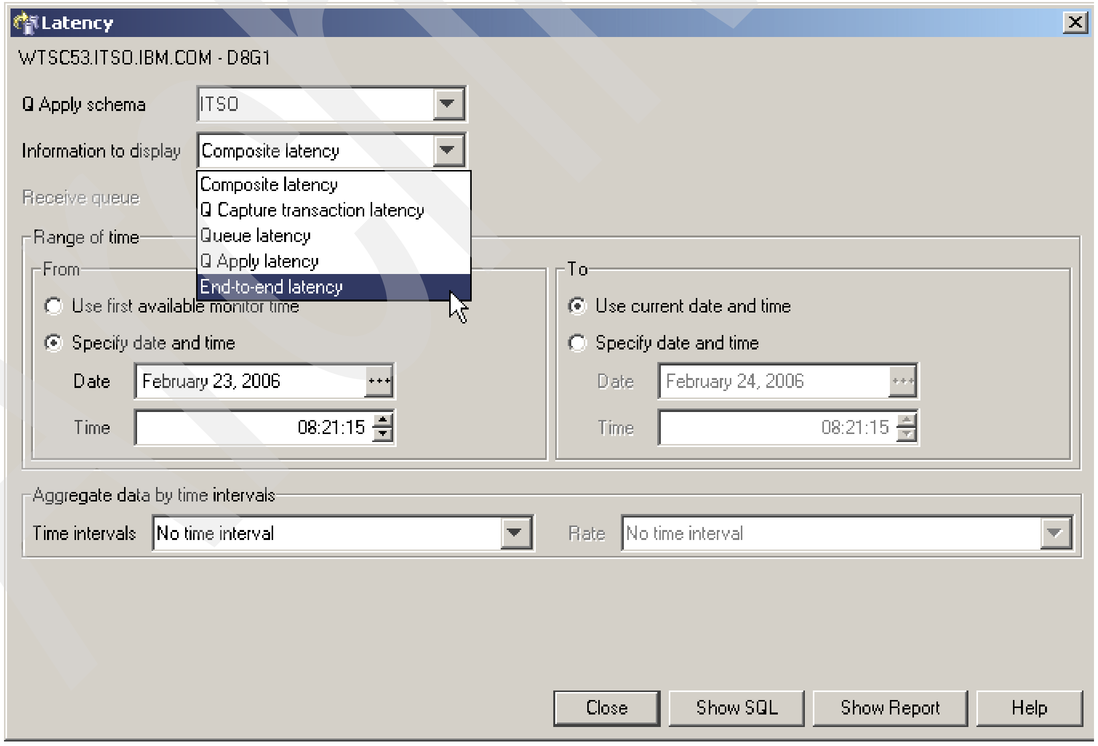
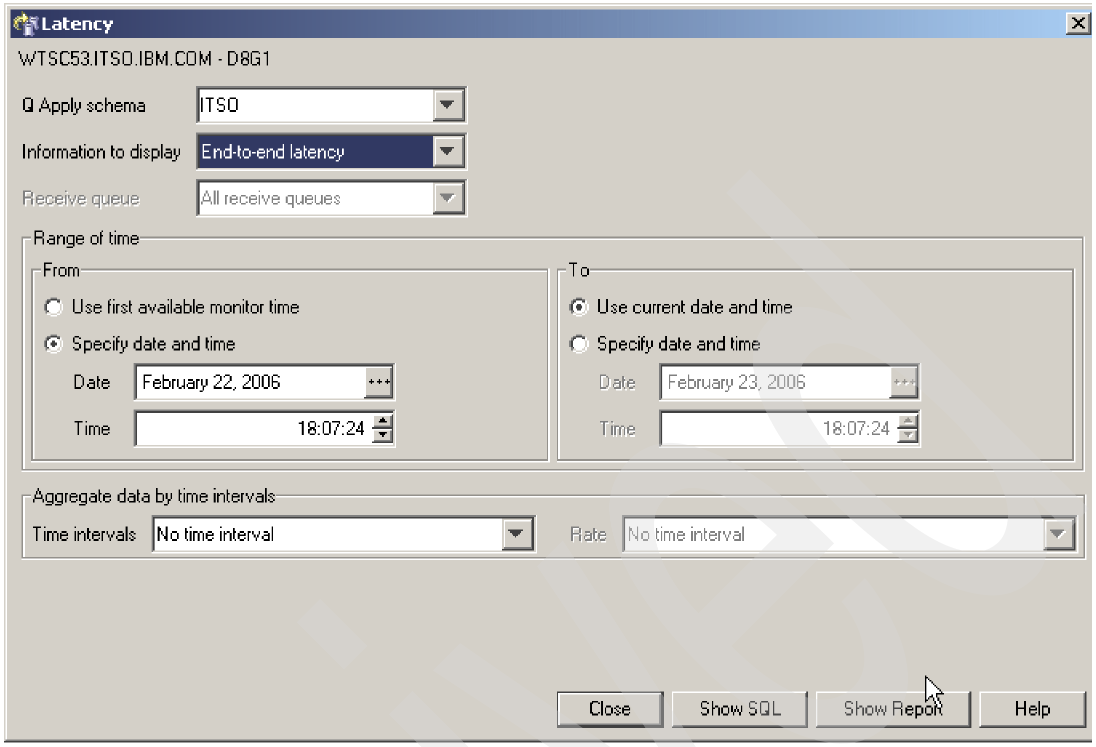


Figure 3-13 Q Capture latency and end-to-end latency using Replication Center 2/4


Latency

WTSC53.ITSO.IBM.COM - D8G1

Q Apply schema: ITSO

Information to display: End-to-end latency

Receive queue: All receive queues

Range of time

From
- ○ Use first available monitor time
- ● Specify date and time
  - Date: February 22, 2006
  - Time: 18:07:24

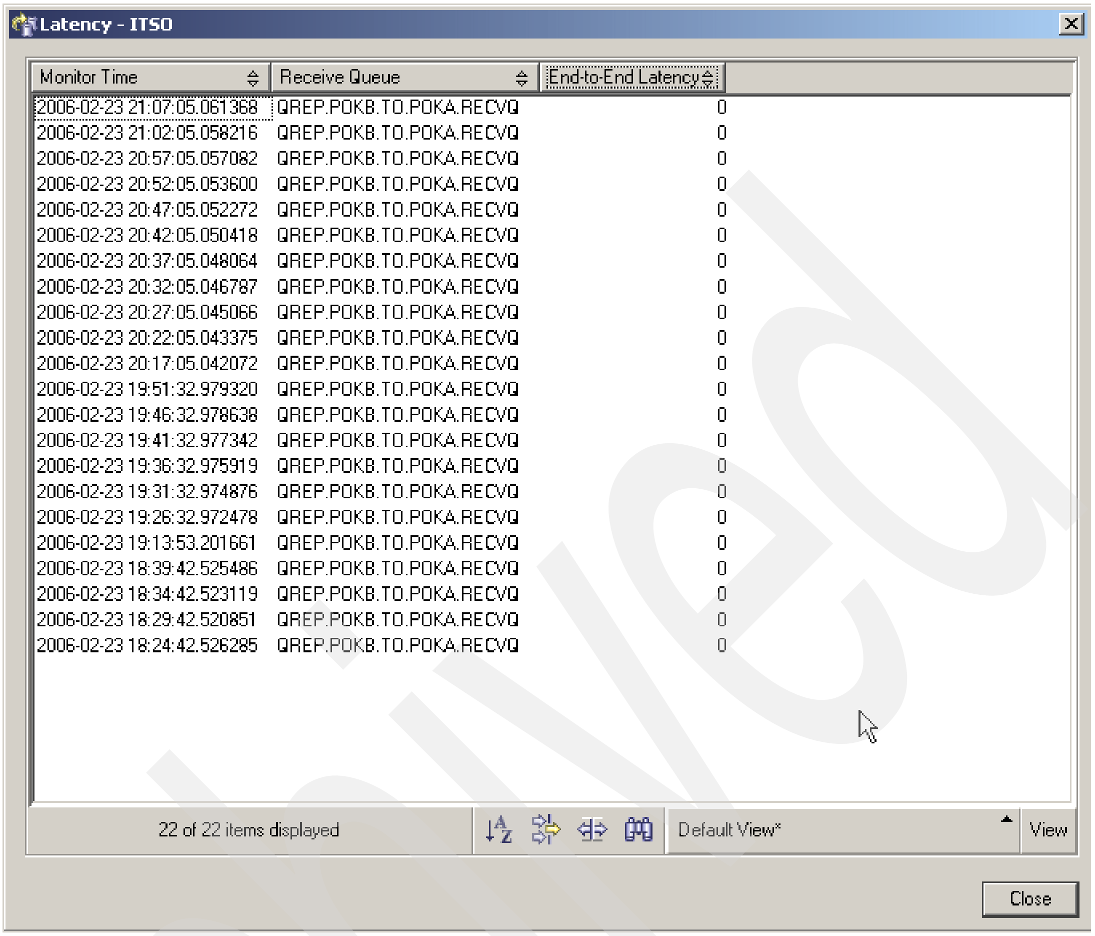
To
- ● Use current date and time
- ○ Specify date and time
  - Date: February 23, 2006
  - Time: 18:07:24

Aggregate data by time intervals

Time intervals: No time interval

Rate: No time interval

Figure 3-14   Q Capture latency and end-to-end latency using Replication Center 3/4



Monitor Time	Receive Queue	End-to-End Latency
2006-02-23 21:07:05.061368	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 21:02:05.058216	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:57:05.057082	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:52:05.053600	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:47:05.052272	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:42:05.050418	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:37:05.048064	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:32:05.046787	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:27:05.045066	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:22:05.043375	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 20:17:05.042072	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:51:32.979320	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:46:32.978638	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:41:32.977342	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:36:32.975919	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:31:32.974876	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:26:32.972478	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 19:13:53.201661	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 18:39:42.525486	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 18:34:42.523119	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 18:29:42.520851	QREP.POKB.TO.POKA.RECVQ	0
2006-02-23 18:24:42.526285	QREP.POKB.TO.POKA.RECVQ	0

22 of 22 items displayed

Figure 3-15 Q Capture latency and end-to-end latency using Replication Center 4/4

Example 3-21 Q Capture latency and end-to-end latency using Live Monitor 1/2

```
-- Live Monitor for Q-Apply command syntax:
-- QApplyMonitor -alias <db2 alias for qapply> -schema <qapply schema>
-- -uid <db2 userid> -pwd <db2 password>
C:\replMonitor>QApplyMonitor -alias DB8A -schema ITSOD -uid hueykim -pwd xxxxxx
```

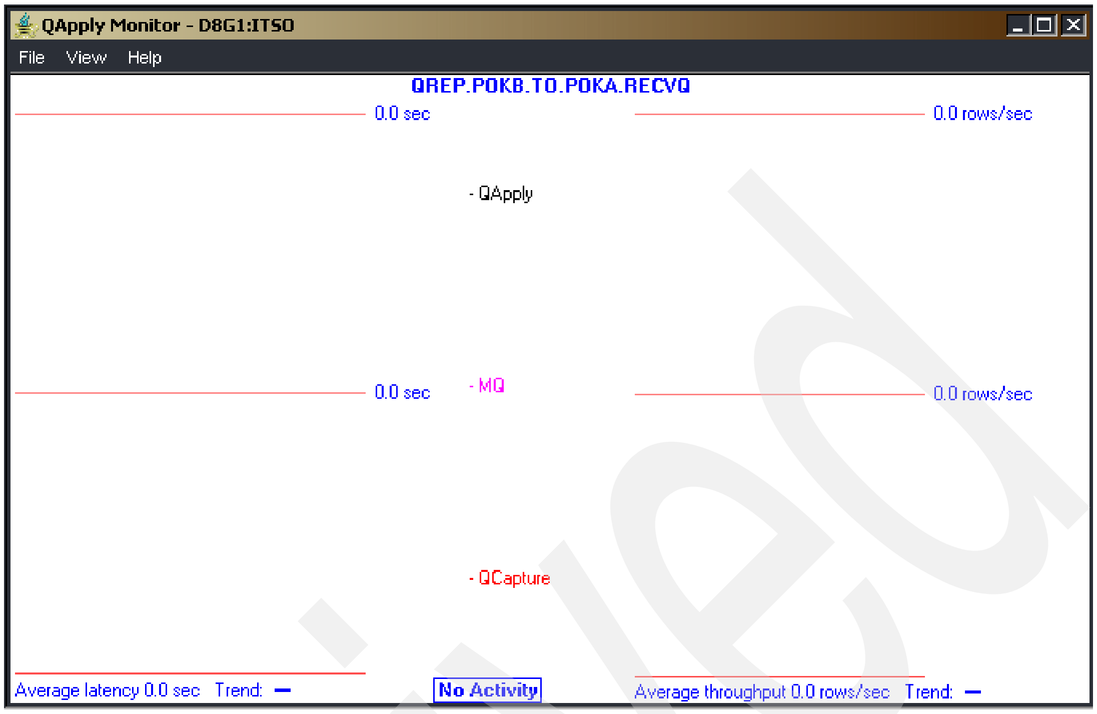


Figure 3-16 Q Capture latency and end-to-end latency using Live Monitor 2/2

STEP SWCF6: Stop workload on secondary

In this step, stop the application workload directed to this server without redirecting it to the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP SWCF7: Resume RO workload on secondary

The read-only workload can now be resumed on the secondary server.

STEP SWCF8: Autostop Q Capture on secondary

In this step, the Q Capture program on the primary server is directed to shut down after it has completed processing all the changes that have been captured while the workload was running, and written it to the send queue.

Example 3-22 shows the command that can be used to achieve this function. To determine whether the Q Capture program has stopped, you should monitor the contents of the Q Capture log, which shows the program as having stopped, similar to that shown in Example 3-3 on page 42.

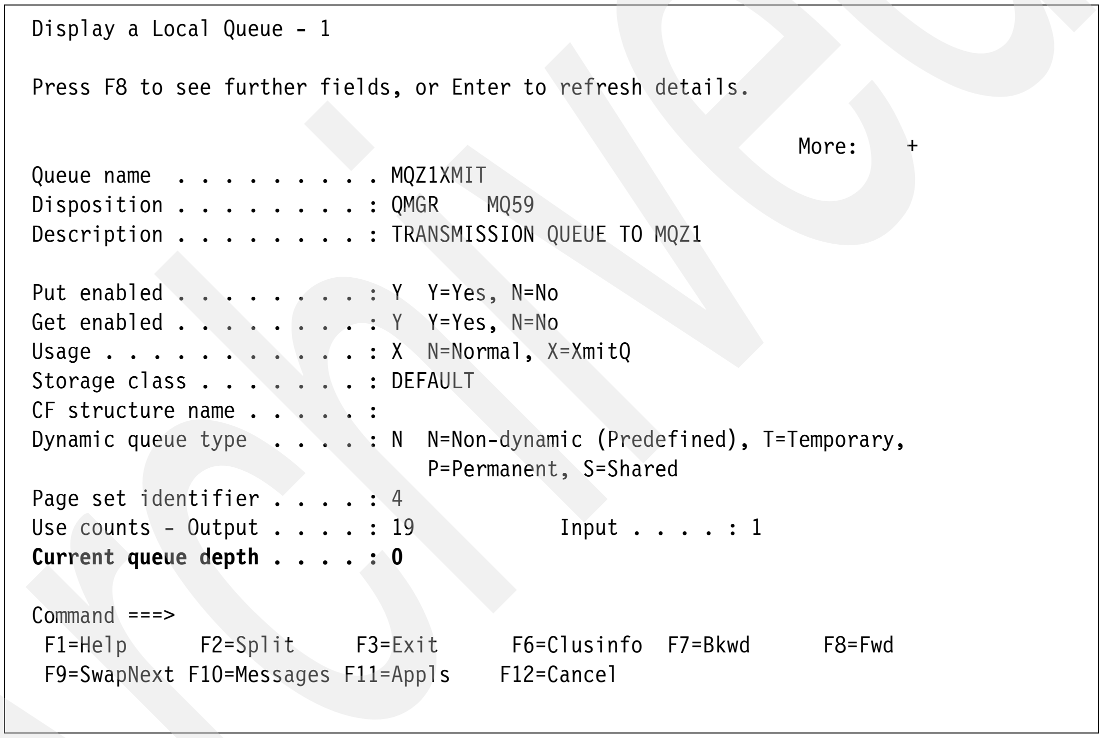
Example 3-22 Autostop Q Capture on the secondary server SC59 - modify command

```
F QCAPITSO, CHGPARMS AUTOSTOP=Y
ASN0523I  "Q Capture" : "ITSO" : "Initial" : The CHGPARMS command
response: "AUTOSTOP" has been set to "Y".
ASN0573I  "Q Capture" : "ITSO" : "Initial" : The program was stopped.
-                                          --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG    PAGE   SWAP    VIO SWAPS
```

```
-QCAPITSO          QCAP        00  13298   8.05    .00   26.7  1119M
 0      0      0      0     0
-QCAPITSO ENDED.  NAME-HUEYKIM             TOTAL CPU TIME=  8.05
TOTAL ELAPSED TIME=  26.7
$HASP395 QCAPITSO ENDED
$HASP309 INIT 2    INACTIVE ******** C=AB
SE '21.25.06 JOB13245 $HASP165 QCAPITSO ENDED AT WTSC59  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

STEP SWCF9: Verify XMITQ on secondary consumed

Before the application workload can be started on the primary server, we need to ensure that all the messages in the transmit queue on the secondary server have been propagated over to the primary server. The value of the Current queue depth field in Figure 3-17 is zero, indicating that the transmit queue MQZ1XMIT on the secondary server has been fully consumed.



```
Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

                                                                More:    +
Queue name  . . . . . . . . . MQZ1XMIT
Disposition . . . . . . . . : QMGR    MQ59
Description . . . . . . . . : TRANSMISSION QUEUE TO MQZ1

Put enabled . . . . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . . . . : Y  Y=Yes, N=No
Usage . . . . . . . . . . . : X  N=Normal, X=XmitQ
Storage class . . . . . . . : DEFAULT
CF structure name . . . . . :
Dynamic queue type  . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                 P=Permanent, S=Shared
Page set identifier . . . . : 4
Use counts - Output . . . . : 19           Input . . . . : 1
Current queue depth . . . . : 0

Command ===>
 F1=Help      F2=Split     F3=Exit      F6=Clusinfo  F7=Bkwd      F8=Fwd
 F9=SwapNext F10=Messages F11=Appls    F12=Cancel
```

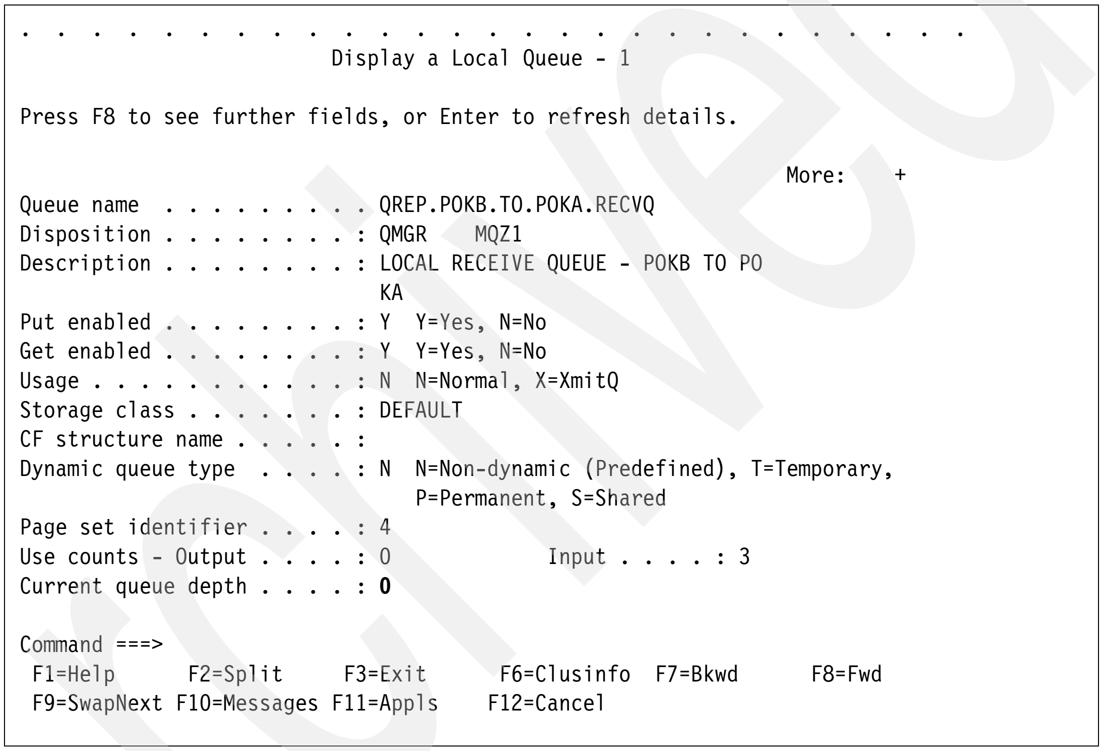
Figure 3-17   Verify transmit queue on the secondary server SC59 is fully consumed

STEP SWCF10: Verify RECVQ on primary consumed

This step verifies that all the messages in the receive queue on the primary server that have been propagated from the secondary server have been consumed before the application workload can be started on the primary server. Doing so eliminates the potential for additional conflicts and lock contention between the Q Apply program and the application workload.

Attention: In order to narrow the window of application workload unavailability, you may choose to start the application workload on the primary server before verifying that the receive queue has been fully consumed. If you choose to do so, conflict exceptions may be generated in addition to potential lock contention. Conflict exceptions in particular will result in older transactions from the secondary server overwriting newer transactions occurring on the primary server. Therefore, we strongly advise against starting the application workload on the primary server before all the messages in the receive queue have been fully consumed.

As described earlier, WebSphere MQ can be used to verify that the value of the Current queue depth field in Figure 3-18 is zero, indicating that the receive queue QREP.POKB.TO.POKA.RECVQ on the primary server has been fully consumed.



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                         Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

                                                              More:    +
Queue name  . . . . . . . . . QREP.POKB.TO.POKA.RECVQ
Disposition . . . . . . . . : QMGR    MQZ1
Description . . . . . . . . : LOCAL RECEIVE QUEUE - POKB TO PO
                              KA
Put enabled . . . . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . . . . : Y  Y=Yes, N=No
Usage . . . . . . . . . . . : N  N=Normal, X=XmitQ
Storage class . . . . . . . : DEFAULT
CF structure name . . . . . :
Dynamic queue type  . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                 P=Permanent, S=Shared
Page set identifier . . . . : 4
Use counts - Output . . . . : 0            Input . . . . : 3
Current queue depth . . . . : 0

Command ===>
 F1=Help      F2=Split     F3=Exit      F6=Clusinfo  F7=Bkwd      F8=Fwd
 F9=SwapNext F10=Messages F11=Appls    F12=Cancel
```

Figure 3-18 Verify receive queue on the primary server SC53 is fully consumed

STEP SWCF11: Resume primary workload

The application workload can now be started on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

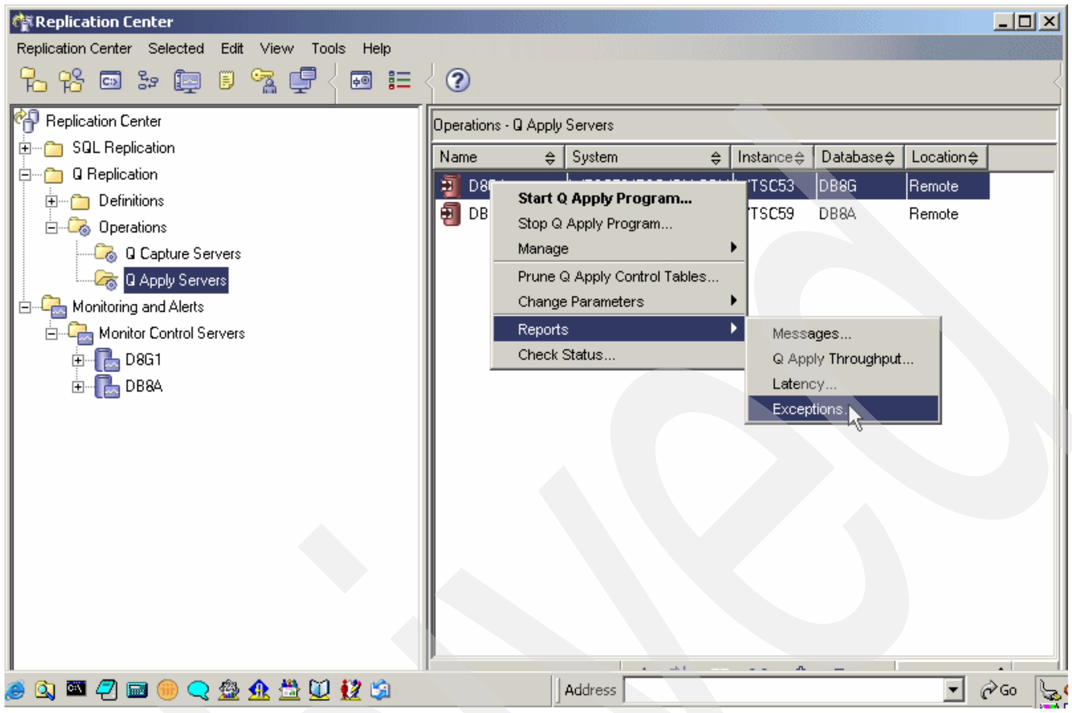
STEP SWCF12: Start Q Capture on secondary

Start Q Capture on the secondary server as described in “STEP SWCF3: Start Q Capture on secondary” on page 50.

STEP SWCF13: Ensure no exceptions recorded

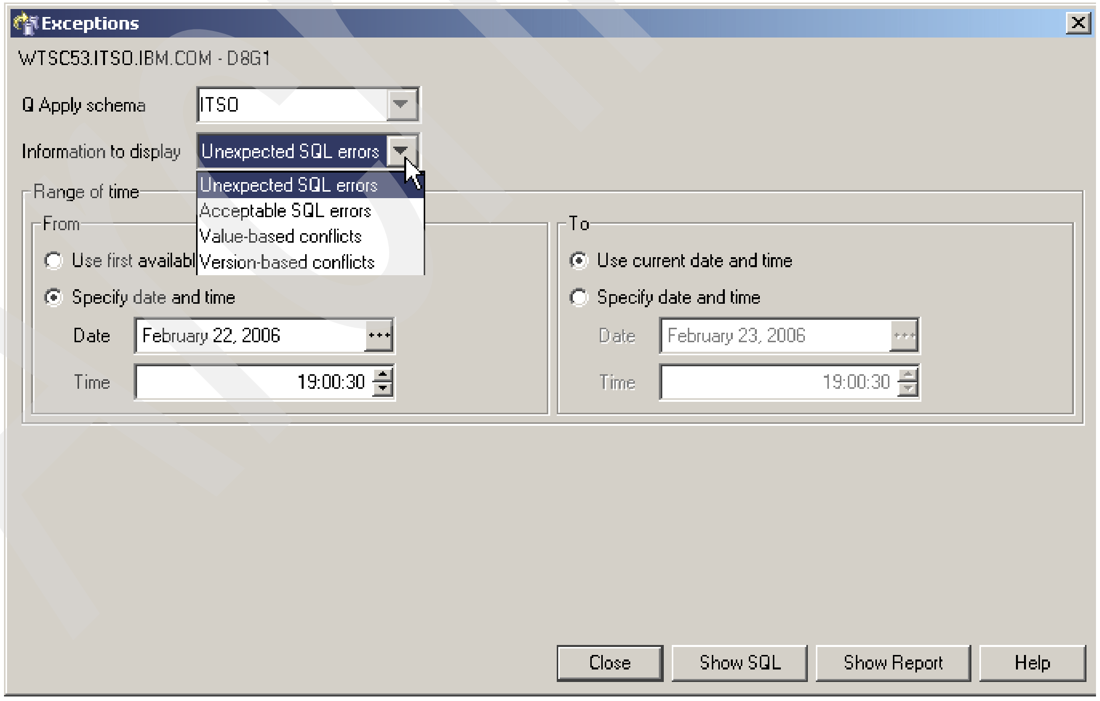
This is a precautionary step since no exceptions should occur on a switchback from a controlled failover. Figure 3-19 through Figure 3-22 on page 60 show the Replication Center

GUI for viewing exceptions. Even though Figure 3-20 shows only "Unexpected SQL errors" being selected, as a precaution all the categories should be selected.



Name	System	Instance	Database	Location
D8		TSC53	DB8G	Remote
DB		TSC59	DB8A	Remote

Figure 3-19 Viewing exceptions through Replication Center 1/4



WTSC53.ITSO.IBM.COM - D8G1

Q Apply schema: ITSO

Information to display: Unexpected SQL errors

From: Use first availabl ( ) [✓ Specify date and time]

Date: February 22, 2006

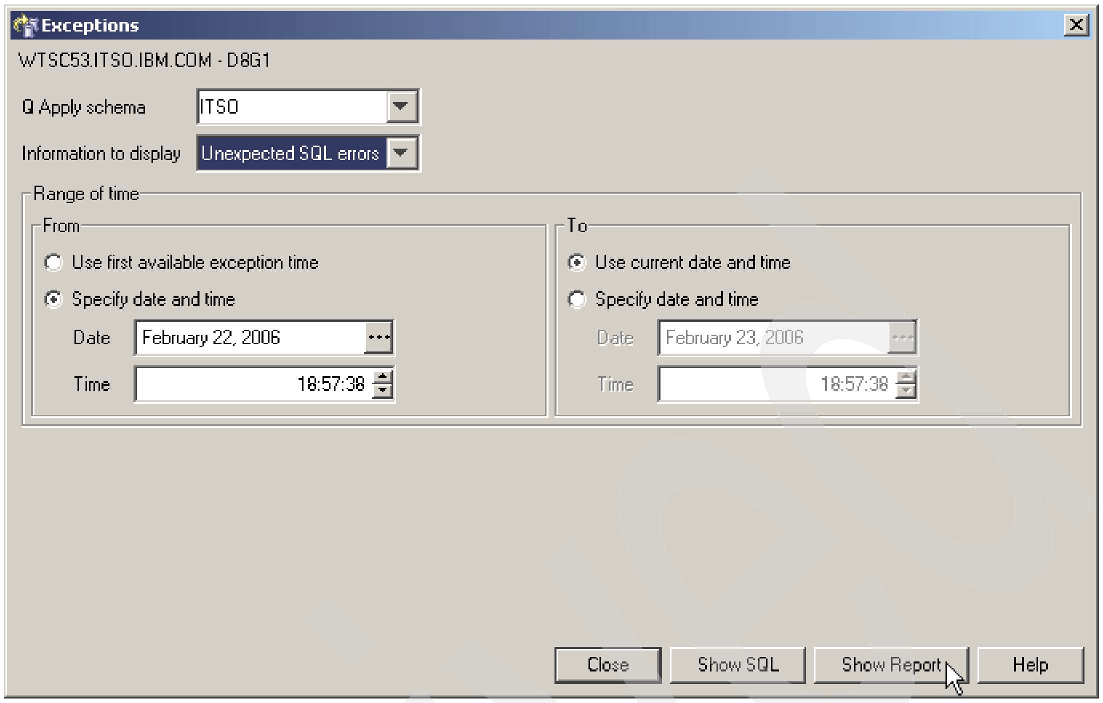
Time: 19:00:30

To: [✓ Use current date and time] Specify date and time ( )

Date: February 23, 2006

Time: 19:00:30

Figure 3-20 Viewing exceptions through Replication Center 2/4


Exceptions

WTSC53.ITSO.IBM.COM - D8G1

Q Apply schema: ITSO

Information to display: Unexpected SQL errors

Range of time

From
- ( ) Use first available exception time
- (●) Specify date and time

Date: February 22, 2006

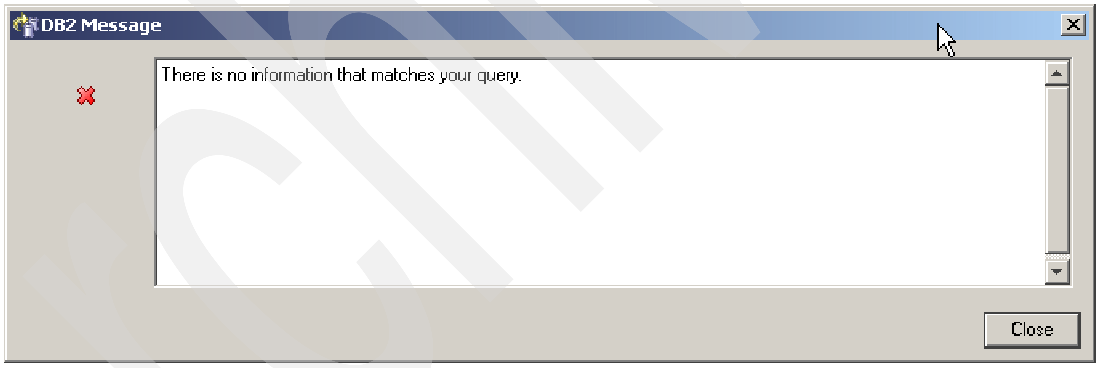
Time: 18:57:38

To
- (●) Use current date and time
- ( ) Specify date and time

Date: February 23, 2006

Time: 18:57:38

Figure 3-21 Viewing exceptions through Replication Center 3/4


DB2 Message

There is no information that matches your query.

Figure 3-22 Viewing exceptions through Replication Center 4/4

3.7 Uncontrolled failover (UF) and switchback

The uncontrolled failover scenario is driven by circumstances that force an organization to transfer the application workload to a standby server. The circumstances may range from a major hardware or software failure that is likely to result in an extended outage of the primary server, to a network failure preventing business-critical applications from connecting to the primary server, thereby necessitating an immediate switchover of the application workload to the standby site. The duration of the outage may be short or extended depending upon the severity of the event that triggered the uncontrolled failover.

Important: With an uncontrolled failover and switchback, data loss and potential conflicts may be unavoidable.

Before “normal”[a] operations can be re-established after an uncontrolled failover, extreme care should be taken to ensure that the data in the primary server and secondary server are back in sync. Multiple options are available to resynchronize the data in the primary and secondary server to achieve this:

1. Identify the mismatching rows when no application workload is running on either the primary server or the secondary server using any one of the following methods:
   - Analyze the exceptions recorded in the IBMQREP_EXCEPTIONS table in the primary and secondary server to identify the mismatching rows. For a detailed understanding of the causes of conflict and SQL error exceptions, especially those that cause data inconsistencies, refer to Appendix B, “Exception processing in a bidirectional Q replication environment” on page 191.
   - Use the ASNTDIFF utility.
2. Rectify the differences using any one of the following methods as appropriate while no application workload is running on either the primary server or the secondary server:
   - Using the ASNTREP utility as follows:
     - Deactivate the subscriptions.
     - Use the ASNTREP utility to resynchronize the data.
     - Reactivate the subscriptions with HAS_LOADPHASE of ‘N’.
     - Reactivate application workload on the primary server.
   - Using SQL updates:
     - Deactivate the subscriptions.
     - Use SQL to update the data in the primary server and the secondary server to bring them back into sync.
     - Optionally (but recommended), use the ANSTDIFF utility to verify that the data is synchronized.
     - Reactivate the subscriptions with HAS_LOADPHASE of ‘N’.
     - Reactivate application workload on the primary server.
   - Resynchronizing the data in the primary server from secondary server or vice versa as follows:
     - Deactivate the subscriptions.
     - Activate the subscriptions with HAS_LOADPHASE of ‘I’ (automatic load), where the source is either the data in the primary server or the secondary server, as the case may be.
     - Reactivate application workload on the primary server.

a. “Normal” operations in a bidirectional high availability environment are where the primary server is running the application workload and the secondary server is running a read-only workload.

The following subsections describe the recommended procedure for an uncontrolled failover, and a choice of multiple procedures for switchback from the uncontrolled failover.

3.7.1 Uncontrolled failover (UF)

The recommended uncontrolled failover procedure is shown in Figure 3-23 and assumes the following:

- Just prior to uncontrolled failover, bidirectional replication is up and running on the primary and secondary server. All subscriptions are active and Q Capture and Q Apply are running on both the primary and secondary servers.
- The secondary server is running a read-only workload at the time of the uncontrolled failover.
- When the decision to initiate the uncontrolled failover procedure is made, it does not necessarily mean that the Q Capture program on the primary server has stopped propagating changes to the secondary server. However, in all likelihood the primary server is totally unavailable and the Q Capture program is no longer able to propagate changes to the secondary server.
- Replication was performing well with the end-to-end latency rate in seconds. It is being monitored and the metrics are acceptable.
- There are no conflict exceptions resulting from this uncontrolled failover on the secondary server since the application workload will only be started after all the messages in the receive queue on the secondary server are consumed.

Attention: The guiding principle of this failover procedure is to minimize the potential for exceptions and contention on the secondary server.

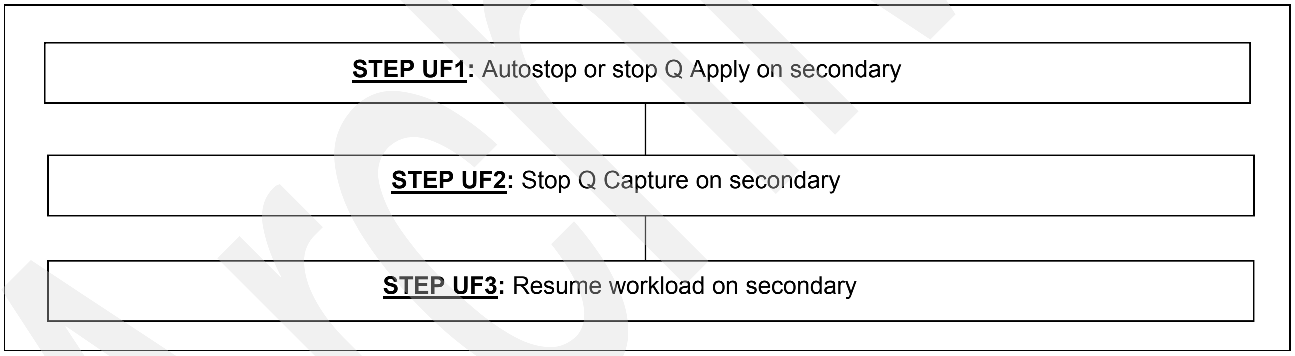

STEP UF1: Autostop or stop Q Apply on secondary
STEP UF2: Stop Q Capture on secondary
STEP UF3: Resume workload on secondary

Figure 3-23 Overview of uncontrolled failover (UF) steps

Each of the steps shown in Figure 3-23 is described briefly in the following sections.

STEP UF1: Autostop or stop Q Apply on secondary

In this step, Q Apply on the secondary server is directed to shut down after it has completed processing all the changes propagated from the primary server, as described in “STEP CF7: Autostop Q Apply on secondary” on page 45.

Attention: If Q Apply does not shut down, it could be because it is has not received the final message of a DB2 transaction and therefore continues to poll the receive queue for it. To determine whether this is the possible cause of Q Apply not shutting down you can:

- Use **`AsnQMFmt`** message formatter to look at the contents of the last message received and see if it had `Last message in DB transaction` in the qMsgHeader.msgFlag field, as shown in Example 3-23. If no such value exists, then the last message received is not the final message of a DB2 transaction, and Q Apply will not shut down.
- Verify that all the messages in the receive queue on the secondary server have been consumed, as described in “STEP UF1a: Verify RECVQ on secondary consumed” on page 64.

If the Current queue depth field remains at not zero for awhile, then an immediate stop of Q Apply must be issued for the Q Apply program, as described in “STEP UF1b: Stop Q Apply on secondary” on page 65.

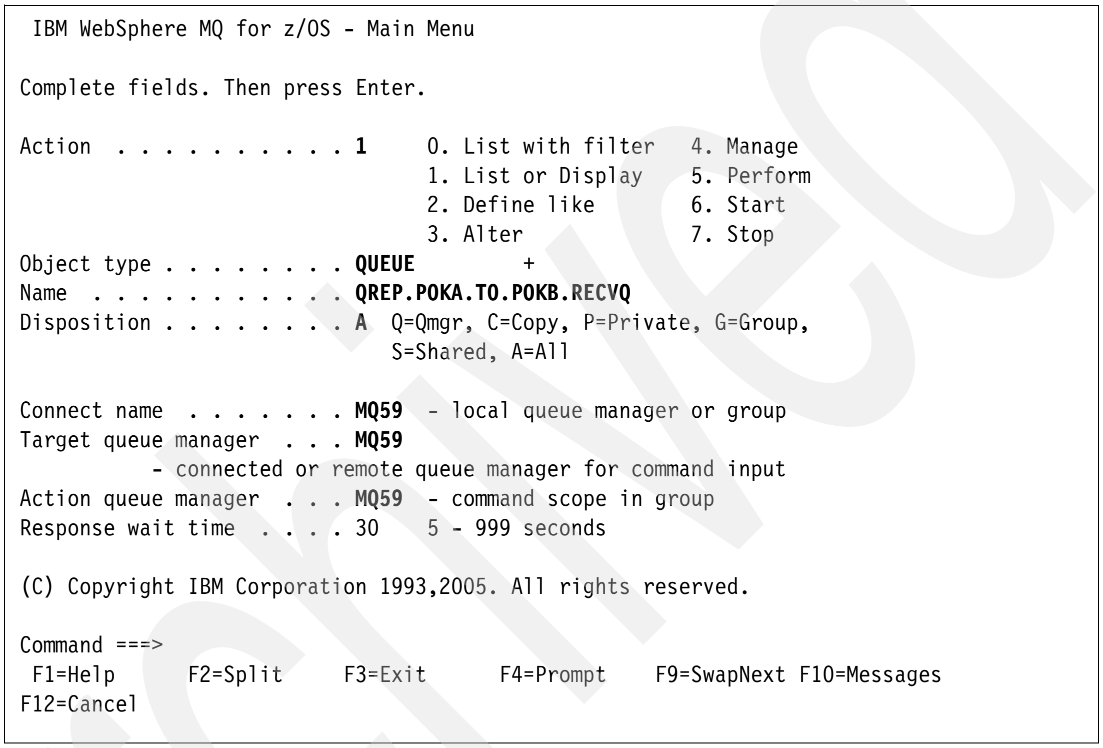
Example 3-23 AsnQMFmt command output with “Last message in DB transaction”

```
--Example of a SQL statement that was run and the corresponding message in the receive
--queue that was displayed by asnqmfmt.
--UPDATE BAL SET BAL_TYPE_CD = 'SR' WHERE ACCT_ID = 1200 ;

  qMsgHeader.msgFamily:  QREP
  qMsgHeader.msgtype:    ASNMQ_TRANS_MSG
  qMsgHeader.msgVersion: ASNMQ_VERSION200
  qMsgHeader.msgFlag:
    Last message in DB transaction.
qTransMsgHeader.segment_num: 0001
qTransMsgHeader.commit_LSN:  0000:0000:0000:2b38:ccdb
qTransMsgHeader.commit_time: 02-21-2006 16:58:51.000001 UTC
qTransMsgHeader.commit_time: 02-21-2006 10:58:51.000001 LOCALTIME
qTransMsgHeader.nRows:       1
qTransMsgHeader.auth_id:       DB2INST9
qTransMsgHeader.auth_tkn:
qTransMsgHeader.plan_id:
qTransMsgHeader.uow_id:        0000:0000:0000:000e:003b
qTransRow.length:      68
rowheader.encodetype:  COMPACT
rowheader.encodever:   ASNMQ_ROW_ENCODE_V100
rowheader.operation:   UPDATE
rowheader.intentseq:   0000:0000:0000:2b38:cc69
rowheader.bafter:      :BEFORE VALUES COLS::AFTER VALUES KEY::AFTER VALUES COLS:
rowheader.sub_id:      1
colset.nCols:   4
colset.length:  16
  col.VALUE:    SUPPRESSED
  col.VALUE:      length:    2
 0000000110122a2a 494E                                       IN
  col.VALUE:    SUPPRESSED
  col.VALUE:    SUPPRESSED
colset.nCols:   1
colset.length:  12
  col.VALUE:      length:    4
 0000000110122a34 000004B0                                   ...°
colset.nCols:   4
colset.length:  16
  col.VALUE:    SUPPRESSED
  col.VALUE:      length:    2
```

```
0000000110122a46 5352                                   SR
 col.VALUE:    SUPPRESSED
 col.VALUE:    SUPPRESSED
```

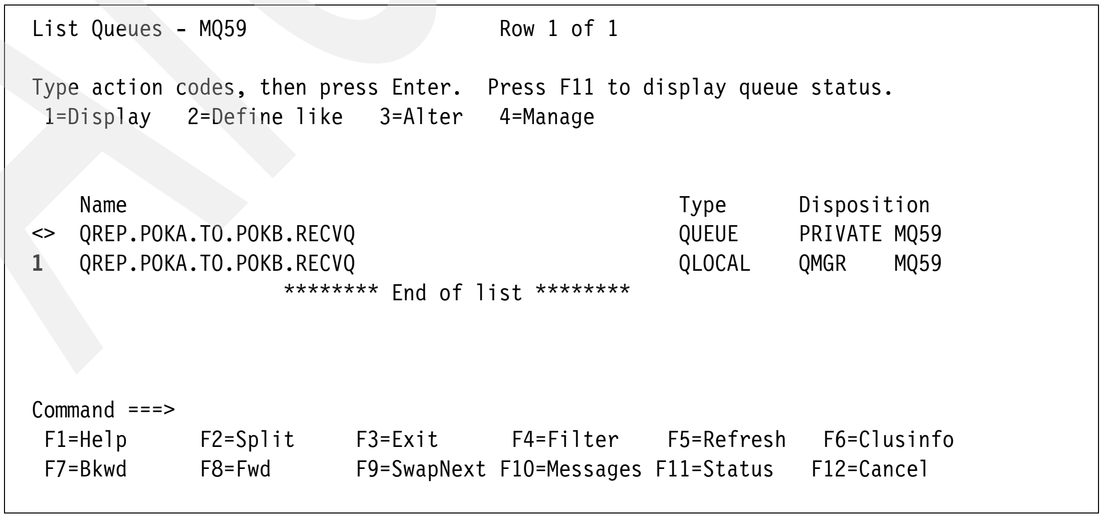
STEP UF1a: Verify RECVQ on secondary consumed

Figure 3-24 through Figure 3-26 show the WebSphere MQ panels to navigate to determine that the receive queue is empty. The value of the Current queue depth field in Figure 3-26 is zero, indicating that the receive queue QREP.POKA.TO.POKB.RECVQ on the secondary server has been fully consumed.



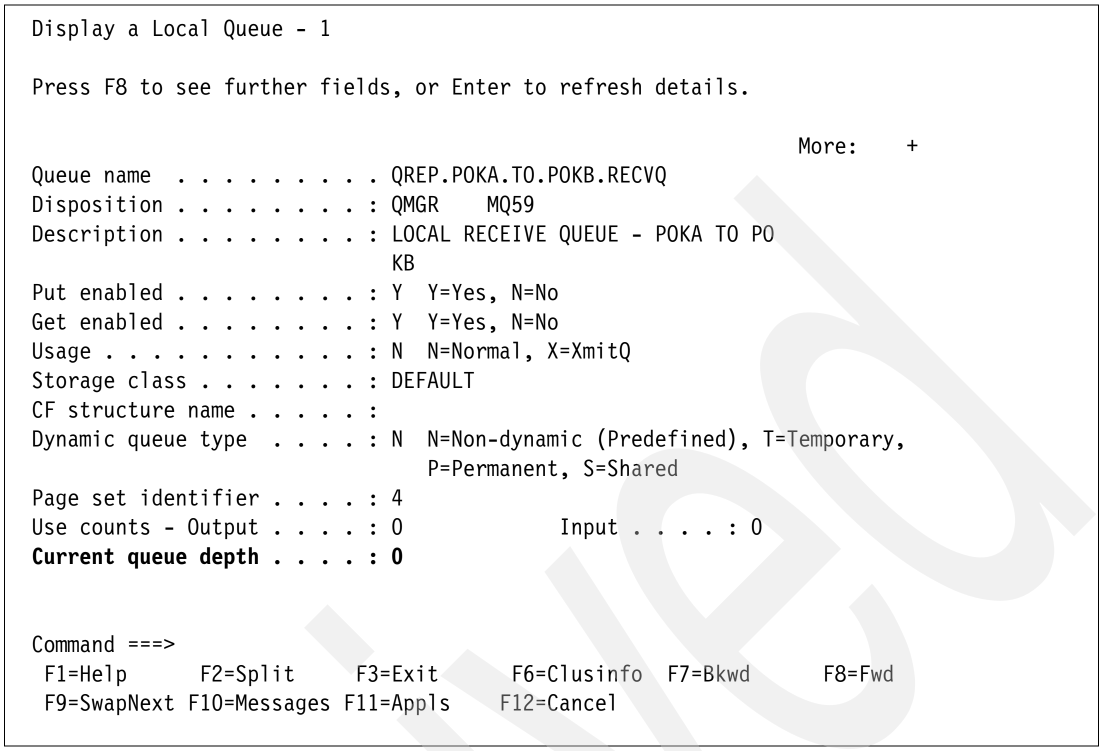
```
 IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . 1     0. List with filter   4. Manage
                                  1. List or Display    5. Perform
                                  2. Define like        6. Start
                                  3. Alter              7. Stop
Object type . . . . . . . . QUEUE         +
Name  . . . . . . . . . . . QREP.POKA.TO.POKB.RECVQ
Disposition . . . . . . . . A  Q=Qmgr, C=Copy, P=Private, G=Group,
                               S=Shared, A=All

Connect name  . . . . . . . MQ59  - local queue manager or group
Target queue manager  . . . MQ59
           - connected or remote queue manager for command input
Action queue manager  . . . MQ59  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Figure 3-24 Verify receive queue on the secondary server SC59 is fully consumed 1/3



```
 List Queues - MQ59                    Row 1 of 1

 Type action codes, then press Enter.  Press F11 to display queue status.
  1=Display   2=Define like   3=Alter   4=Manage


     Name                                        Type      Disposition
 <>  QREP.POKA.TO.POKB.RECVQ                     QUEUE     PRIVATE MQ59
 1   QREP.POKA.TO.POKB.RECVQ                     QLOCAL    QMGR    MQ59
                      ******** End of list ********



 Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F6=Clusinfo
  F7=Bkwd      F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 3-25 Verify receive queue on the secondary server SC59 is fully consumed 2/3


```
Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

                                                              More:    +
Queue name  . . . . . . . . . QREP.POKA.TO.POKB.RECVQ
Disposition . . . . . . . . : QMGR    MQ59
Description . . . . . . . . : LOCAL RECEIVE QUEUE - POKA TO PO
                              KB
Put enabled . . . . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . . . . : Y  Y=Yes, N=No
Usage . . . . . . . . . . . : N  N=Normal, X=XmitQ
Storage class . . . . . . . : DEFAULT
CF structure name . . . . . :
Dynamic queue type  . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                 P=Permanent, S=Shared
Page set identifier . . . . : 4
Use counts - Output . . . . : 0             Input . . . . : 0
Current queue depth . . . . : 0


Command ===>
 F1=Help      F2=Split     F3=Exit      F6=Clusinfo  F7=Bkwd      F8=Fwd
 F9=SwapNext F10=Messages F11=Appls    F12=Cancel
```
Figure 3-26 Verify receive queue on the secondary server SC59 is fully consumed 3/3

STEP UF1b: Stop Q Apply on secondary

Example 3-24 shows the command that can be used to stop Q Apply on the secondary server. To determine whether the Q Apply program has stopped, you should monitor the contents of the Q Apply log, which shows the program as having stopped, as shown in Example 3-25.

Example 3-24 Stop Q Apply on the secondary server SC59 - modify command

```
F QAPPITSO, STOP
ASN0522I  "Q Apply" : "ITSO" : "Initial" : The program received the
"STOP" command.
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QAPPITSO          QAPP        00  20197    .05    .00   40.1  9630K
 0      0      0      0     0
-QAPPITSO ENDED.  NAME-HUEYKIM             TOTAL CPU TIME=   .05
TOTAL ELAPSED TIME=  40.1
$HASP395 QAPPITSO ENDED
$HASP309 INIT 2    INACTIVE ******** C=AB
SE '13.11.37 JOB13271 $HASP165 QAPPITSO ENDED AT WTSC59  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

Example 3-25 Stop Q Apply log contents on the secondary server SC59

```
................
2006-02-24-13.11.29.908645 <handleZOsEvent> ASN0522I  "Q Apply" : "ITSO" : "Initial" : The
program received the "STOP" command.
..........................
```

```
2006-02-24-13.11.34.916418 <asnqapp::main> ASN0573I  "Q Apply" : "ITSO" : "Initial" : The
program was stopped.
```

STEP UF2: Stop Q Capture on secondary

In this step, an immediate stop is issued for the Q Capture program on the secondary server, as described in “STEP CF3: Stop Q Capture on secondary” on page 42.

STEP UF3: Resume workload on secondary

The application workload can now be started on the secondary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

3.7.2 Option A: Automatically resync primary and secondary servers

Table 3-1 on page 37 summarizes the considerations when choosing between four possible switchback options. Option A described here involves starting Q Apply and warm starting Q Capture on the primary server when it becomes available. No data loss is incurred in this scenario.

With an uncontrolled failover, the procedure for switching back to the primary server and establishing normal operations is far more complex than that of a switchback after a controlled failover.

The option A switchback procedure after a successful uncontrolled failover is shown in Figure 3-27 and assumes the following:

- The primary server is up and running DB2.
- WebSphere MQ is up and running on both the primary and secondary servers.
- The primary server is not running any workload—read-only or update.
- Q Capture and Q Apply are not running on either the primary or secondary servers.
- All the subscriptions are in an active state.
- The application workload is running on the secondary server, causing changes to be written to the DB2 log.
- Switchback is performed during a maintenance period when the throughput is low.
- Potential conflict and SQL error exceptions are recorded in the IBMQREP_EXCEPTIONS table on the primary and secondary server. These exceptions are likely to occur during switchback because of unpropagated changes from the primary server to the secondary server, and vice versa.

Attention: The guiding principle of this switchback procedure is to:

- Minimize the potential for exceptions and contention on the primary and secondary server.
- Narrow the window of application workload unavailability by stopping the workload as late as possible and restarting it as early as possible.

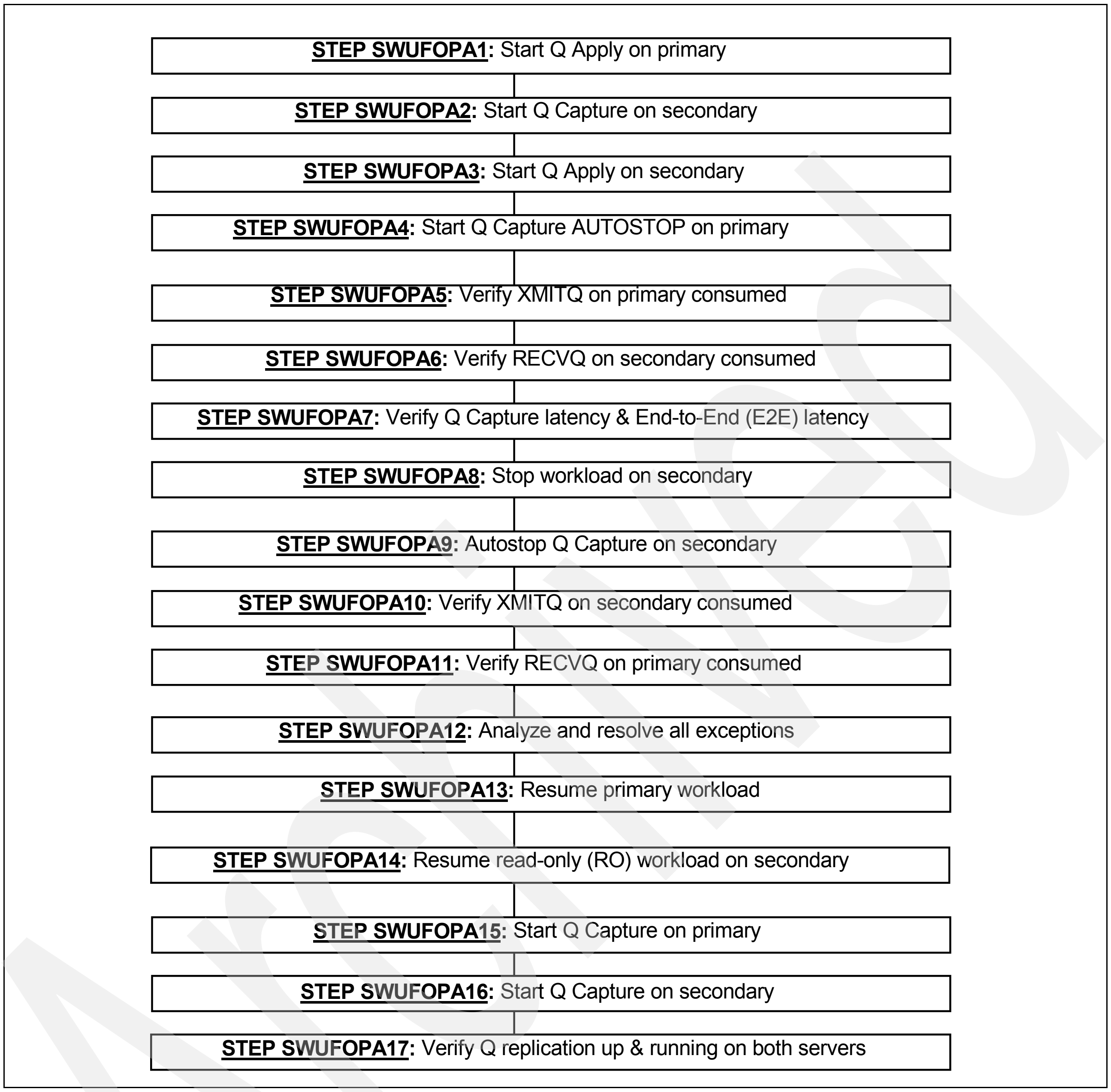

STEP SWUFOPA1: Start Q Apply on primary
STEP SWUFOPA2: Start Q Capture on secondary
STEP SWUFOPA3: Start Q Apply on secondary
STEP SWUFOPA4: Start Q Capture AUTOSTOP on primary
STEP SWUFOPA5: Verify XMITQ on primary consumed
STEP SWUFOPA6: Verify RECVQ on secondary consumed
STEP SWUFOPA7: Verify Q Capture latency & End-to-End (E2E) latency
STEP SWUFOPA8: Stop workload on secondary
STEP SWUFOPA9: Autostop Q Capture on secondary
STEP SWUFOPA10: Verify XMITQ on secondary consumed
STEP SWUFOPA11: Verify RECVQ on primary consumed
STEP SWUFOPA12: Analyze and resolve all exceptions
STEP SWUFOPA13: Resume primary workload
STEP SWUFOPA14: Resume read-only (RO) workload on secondary
STEP SWUFOPA15: Start Q Capture on primary
STEP SWUFOPA16: Start Q Capture on secondary
STEP SWUFOPA17: Verify Q replication up & running on both servers

Figure 3-27 Overview of Option A steps

Each of the steps shown in Figure 3-27 is described briefly in the following sections.

STEP SWUFOPA1: Start Q Apply on primary

Start Q Apply on the primary server, as described in “STEP SWCF1: Start Q Apply on primary” on page 47.

STEP SWUFOPA2: Start Q Capture on secondary

Start Q Capture on the secondary server, as described in “STEP SWCF3: Start Q Capture on secondary” on page 50.

STEP SWUFOPA3: Start Q Apply on secondary

Start Q Apply on the secondary server as described in “STEP SWCF4: Start Q Apply on secondary” on page 51.

STEP SWUFOPA4: Start Q Capture AUTOSTOP on primary

In this step, start Q Capture on the primary server in AUTOSTOP mode by submitting the job shown in Example 3-26, so that it shuts down after it has completed processing all the committed changes that have been captured while the workload was running on the primary server, and written them to the send queue. You can monitor the contents of the Q Capture log, which shows the Q Capture program initializing successfully, similar to that shown in Example 3-16 on page 50.

Example 3-26 JCL to start Q Capture AUTOSTOP on the primary server SC53

```
//QCAPITSO JOB NOTIFY=&SYSUID,                                         JOB17436
//         MSGCLASS=H,MSGLEVEL=(1,0),
//         REGION=OM,TIME=NOLIMIT
/*JOBPARM S=SC53
//QCAP     EXEC PGM=ASNQCAP,
// PARM='/CAPTURE_SERVER=D8G1 capture_schema=ITSO startmode=warmsi
//             AUTOSTOP=Y CAPTURE_PATH=//''HUEYKIM'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//         DD DSN=MQ600.SCSQANLE,DISP=SHR
//         DD DSN=MQ600.SCSQLOAD,DISP=SHR
//*        DD DSN=SYS!!O.SCEERUN,DISP=SHR
//*        DD DSN=SYS!!O.SCLBDLL,DISP=SHR
//*        DD DSN=XML!!O.SIXMMOD1,DISP=SHR
//*TRCIN    DD  DISP=SHR,DSN=USRT001.PARMLIB(TRCIN),
//*             VOL=SER=DRRSHR,UNIT=SYSDA
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,70)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

STEP SWUFOPA5: Verify XMITQ on primary consumed

Verify that the transmit queue on the primary server is fully consumed, as described in “STEP CF5: Verify XMITQ on primary consumed” on page 44.

STEP SWUFOPA6: Verify RECVQ on secondary consumed

Verify that the receive queue on the secondary server is fully consumed, as described in “STEP UF1a: Verify RECVQ on secondary consumed” on page 64.

STEP SWUFOPA7: Verify Q Capture latency and E2E latency

In order to minimize the window during which the business application is not available to end users, it is desirable to shut it down on the secondary server after most of the unpropagated changes have been replicated over to the primary server as described in “STEP SWCF5: Verify Q Capture latency and E2E latency” on page 51.

STEP SWUFOPA8: Stop workload on secondary

In this step, stop the application workload directed to this server without redirecting it to the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP SWUFOPA9: Autostop Q Capture on secondary

In this step, the Q Capture program on the secondary server is directed to shut down after it has completed processing all the committed changes that have been captured while the workload was running on the primary server, and written them to the send queue, as described in "STEP SWCF8: Autostop Q Capture on secondary" on page 56.

STEP SWUFOPA10: Verify XMITQ on secondary consumed

Verify that the transmit queue on the secondary server is fully consumed, as described in "STEP SWCF9: Verify XMITQ on secondary consumed" on page 57.

STEP SWUFOPA11: Verify RECVQ on primary consumed

Verify that the receive queue on the primary server is fully consumed, as described in "STEP SWCF10: Verify RECVQ on primary consumed" on page 57.

STEP SWUFOPA12: Analyze and resolve all exceptions

Exceptions may be generated in the IBMQREP_EXCEPTIONS table at the primary and secondary servers. A careful analysis of these exceptions must be conducted and resolved so that data in the primary and secondary servers are fully synchronized. For a detailed understanding of the causes of conflict and SQL error exceptions, especially those that cause data inconsistencies, refer to Appendix B, "Exception processing in a bidirectional Q replication environment" on page 191.

Important: If the data in the primary and secondary server are not brought into sync before the application workload is started on the primary server, the data content on the two servers will quickly diverge and may therefore become unusable.

STEP SWUFOPA13: Resume primary workload

The application workload can now be started on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

STEP SWUFOPA14: Resume RO workload on secondary

The read-only workload can now be resumed on the secondary server.

STEP SWUFOPA15: Start Q Capture on primary

Start Q Capture on the primary server, as described in "STEP SWCF2: Start Q Capture on primary" on page 49.

STEP SWUFOPA16: Start Q Capture on secondary

Start Q Capture on the secondary server, as described in "STEP SWCF3: Start Q Capture on secondary" on page 50.

STEP SWUFOPA17: Verify that Q replication up and running

Verify that normal operations have been re-established as follows:

- By checking the status of Q Capture and Q Apply on the primary and secondary servers by issuing the **asnqccmd** and **asnqacmd** commands, respectively, as shown in Example 3-27 through Example 3-30 on page 71.
- By checking that all the subscription states are active by issuing SQL statements against the IBMQREP_SUBS tables on the primary and secondary servers, as shown in Example 3-31 on page 71 through Example 3-32 on page 72.

Example 3-27 Verify Q Replication is up and running on the primary and secondary servers 1/6

```
EZYTE27I login: hueykim
HUEYKIM @ SC53:/u/hueykim>asnqccmd capture_server=D8G1 capture_schema=ITSO status
2006-02-24-14.22.24.630386 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x123d9af000000000, tcb = 006FD098
2006-02-24-14.22.26.658432 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-14.22.26.660321 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
2006-02-24-14.22.26.662368 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "PruneThread" thread is in the "is resting" state.
2006-02-24-14.22.26.663731 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "WorkerThread" thread is in the "is doing work" state.
2006-02-24-14.22.26.739903 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "txrdThread" thread is in the "is doing work" state.
HUEYKIM @ SC53:/u/hueykim>
```

Example 3-28 Verify Q Replication is up and running on the primary and secondary servers 2/6

```
HUEYKIM @ SC53:/u/hueykim>asnqacmd apply_server=D8G1 apply_schema=ITSO status
2006-02-24-14.27.35.577464 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x123d9af000000000, tcb = 006FD098
2006-02-24-14.27.37.601382 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-14.27.37.604346 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
2006-02-24-14.27.37.605741 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "MonitorThread" thread is in the "is resting" state.
2006-02-24-14.27.37.608405 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "BR00000" thread is in the "is doing work" state.
```

Example 3-29 Verify Q Replication is up and running on the primary and secondary servers 3/6

```
HUEYKIM@SC59:/u/hueykim> asnqccmd capture_server=DB8A capture_schema=ITSO status
2006-02-24-14.29.11.059295 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x1fe1aa4000000000, tcb = 009E6CF0
2006-02-24-14.29.13.078903 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-14.29.13.079907 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
2006-02-24-14.29.13.080489 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "PruneThread" thread is in the "is resting" state.
2006-02-24-14.29.13.081232 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "WorkerThread" thread is in the "is doing work" state.
2006-02-24-14.29.13.099903 ASN0520I  "AsnQCcmd" : "ITSO" : "Initial" : The STATUS command
response: "txrdThread" thread is in the "is doing work" state.
```

Example 3-30 Verify Q Replication is up and running on the primary and secondary servers 4/6

```
HUEYKIM@SC59:/u/hueykim> asnqacmd apply_server=DB8A apply_schema=ITSO status
2006-02-24-14.30.23.959947 ASN8999D  "N/A" : "N/A" : "Initial" : thread name = Initial, tid
= 0x1fe1aa4000000000, tcb = 009FD098
2006-02-24-14.30.25.978492 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-14.30.25.979502 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "AdminThread" thread is in the "is resting" state.
2006-02-24-14.30.25.980431 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "MonitorThread" thread is in the "is resting" state.
2006-02-24-14.30.25.981034 ASN0520I  "AsnQAcmd" : "ITSO" : "Initial" : The STATUS command
response: "BR00000" thread is in the "is doing work" state.
```

Example 3-31 Verify Q Replication is up and running on the primary and secondary servers 5/6

```
D:\SG247215\Code>db2 -tvf check_source_state.sql
connect to d8g1 user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = D8G1


select subname, state from itso.ibmqrep_subs order by subname

SUBNAME
                                                     STATE
--------------------------------------------------------------------------------
---------------------------------------------------- -----
BAL_V10002
                                                     A
OPTIONS_V10002
                                                     A
PROD_V10002
                                                     A
TRAN_V10002
                                                     A

  4 record(s) selected.


select subname, state from itso.ibmqrep_targets order by subname

SUBNAME
                                                     STATE
--------------------------------------------------------------------------------
---------------------------------------------------- -----
BAL_V10001
                                                     A
OPTIONS_V10001
                                                     A
PROD_V10001
                                                     A
TRAN_V10001
                                                     A

  4 record(s) selected.
```

```
terminate
DB20000I  The TERMINATE command completed successfully.
```

Example 3-32 Verify Q Replication is up and running on the primary and secondary servers 6/6

```
D:\SG247215\Code>db2 -tvf check_target_state.sql
connect to db8a user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = DB8A


select subname, state from itso.ibmqrep_subs order by subname

SUBNAME
                                                     STATE
-------------------------------------------------------------------------------
---------------------------------------------------- -----
BAL_V10001
                                                     A
OPTIONS_V10001
                                                     A
PROD_V10001
                                                     A
TRAN_V10001
                                                     A

  4 record(s) selected.

select subname, state from itso.ibmqrep_targets order by subname

SUBNAME
                                                     STATE
-------------------------------------------------------------------------------
---------------------------------------------------- -----
BAL_V10002
                                                     A
OPTIONS_V10002
                                                     A
PROD_V10002
                                                     A
TRAN_V10002
                                                     A

  4 record(s) selected.

terminate
DB20000I  The TERMINATE command completed successfully.
```

3.7.3 Option B: Resync primary and reinitialize secondary

Table 3-1 on page 37 summarizes the considerations when choosing between four possible switchback options. Option B involves starting Q Apply on the primary server when it becomes available, but not Q Capture on the primary server right away. It involves re-synchronizing the primary server with unpropagated changes from the secondary server.

Once that is completed, the subscriptions are deactivated and reactivated so that the secondary server is re-initialized with the contents of the primary server.

With an uncontrolled failover, the procedure for switching back to the primary server and establishing normal operations is far more complex than that of a switchback after a controlled failover.

The option B switchback procedure after a successful uncontrolled failover is shown in Figure 3-28 and assumes the following:

- The primary server is up and running DB2.
- WebSphere MQ is up and running on both the primary and secondary servers.
- The primary server is not running any workload—read-only or update.
- Q Capture and Q Apply are not running on either the primary or secondary servers.
- All the subscriptions are in an active state.
- The application workload is running on the secondary server, causing changes to be written to the DB2 log.
- Switchback is performed during a maintenance period when the throughput is low.
- Potential conflict and SQL error exceptions are recorded in the IBMQREP_EXCEPTIONS table on the primary server. There are no exceptions at the secondary server because unpropagated changes from the primary server to the secondary server are ignored in this scenario. These exceptions are likely to occur during switchback because of unpropagated changes from the primary server to the secondary server.

Attention: The guiding principle of this switchback procedure is to:

- Minimize the potential for exceptions and contention on the primary server.
- Narrow the window of application workload unavailability by stopping the workload as late as possible and restarting it as early as possible.

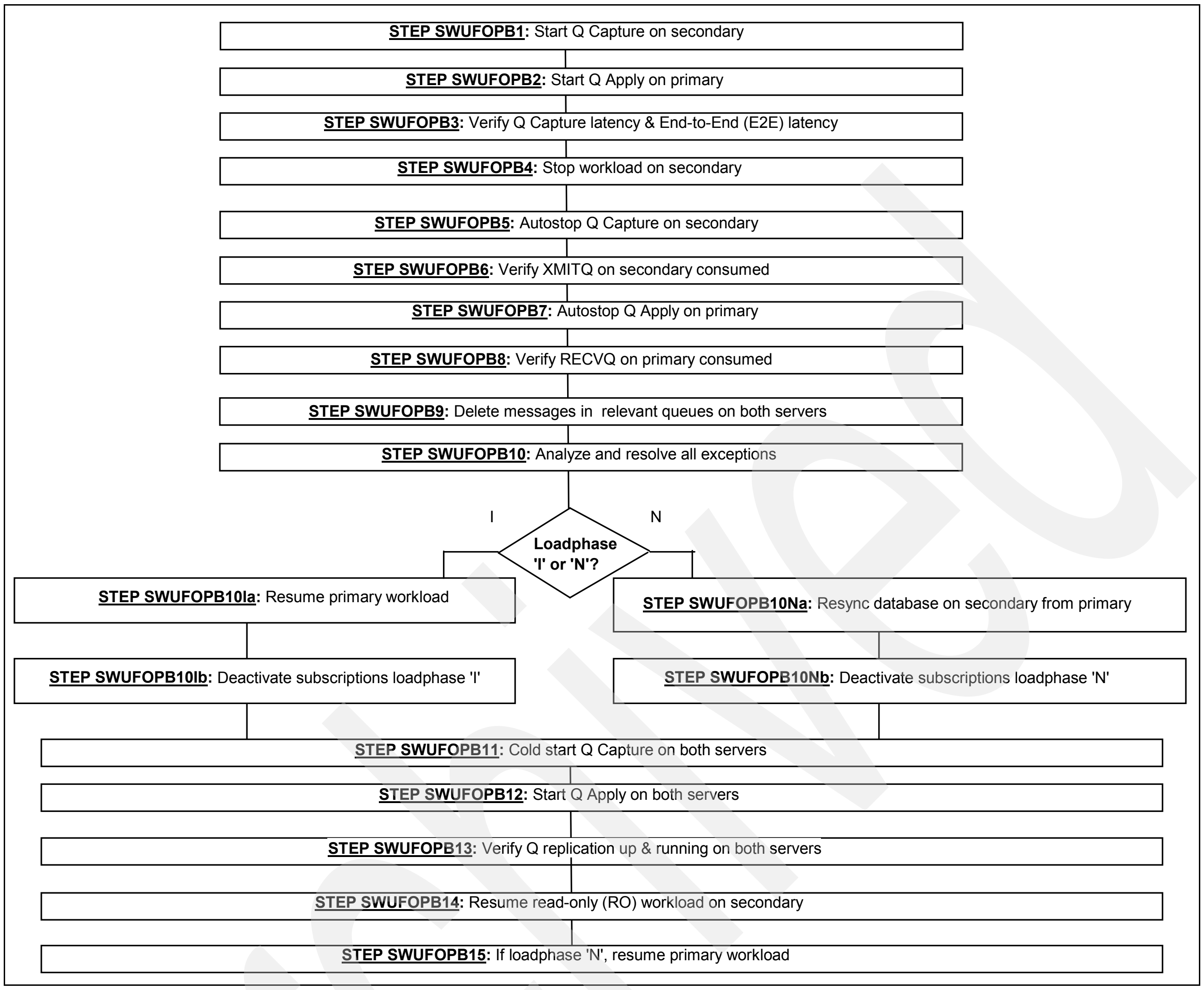

STEP SWUFOPB1: Start Q Capture on secondary
STEP SWUFOPB2: Start Q Apply on primary
STEP SWUFOPB3: Verify Q Capture latency & End-to-End (E2E) latency
STEP SWUFOPB4: Stop workload on secondary
STEP SWUFOPB5: Autostop Q Capture on secondary
STEP SWUFOPB6: Verify XMITQ on secondary consumed
STEP SWUFOPB7: Autostop Q Apply on primary
STEP SWUFOPB8: Verify RECVQ on primary consumed
STEP SWUFOPB9: Delete messages in relevant queues on both servers
STEP SWUFOPB10: Analyze and resolve all exceptions
I
N
Loadphase 'I' or 'N'?
STEP SWUFOPB10Ia: Resume primary workload
STEP SWUFOPB10Na: Resync database on secondary from primary
STEP SWUFOPB10Ib: Deactivate subscriptions loadphase 'I'
STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'
STEP SWUFOPB11: Cold start Q Capture on both servers
STEP SWUFOPB12: Start Q Apply on both servers
STEP SWUFOPB13: Verify Q replication up & running on both servers
STEP SWUFOPB14: Resume read-only (RO) workload on secondary
STEP SWUFOPB15: If loadphase 'N', resume primary workload

Figure 3-28 Overview of Option B steps

Each of the steps shown in Figure 3-28 is described briefly in the following sections.

STEP SWUFOPB1: Start Q Capture on secondary

Start Q Capture on the secondary server, as described in “STEP SWCF3: Start Q Capture on secondary” on page 50.

STEP SWUFOPB2: Start Q Apply on primary

Start Q Apply on the primary server, as described in “STEP SWCF1: Start Q Apply on primary” on page 47.

STEP SWUFOPB3: Verify Q Capture latency and E2E latency

In order to minimize the window during which the business application is not available to end users, it is desirable to shut it down on the secondary server after most of the unpropagated changes have been replicated over to the primary server. “STEP SWCF5: Verify Q Capture latency and E2E latency” on page 51 describes the procedure for determining the Q Capture and end-to-end latency.

STEP SWUFOPB4: Stop workload on secondary

In this step, stop the application workload directed to this server without redirecting it to the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP SWUFOPB5: Autostop Q Capture on secondary

In this step, the Q Capture program on the secondary server is directed to shut down after it has completed processing all the committed changes that have been captured while the workload was running on the primary server, and written them to the send queue, as described in "STEP SWCF8: Autostop Q Capture on secondary" on page 56.

STEP SWUFOPB6: Verify XMITQ on secondary consumed

Verify that the transmit queue on the secondary server is fully consumed, as described in "STEP SWCF9: Verify XMITQ on secondary consumed" on page 57.

STEP SWUFOPB7: Autostop Q Apply on primary

In this step, Q Apply on the primary server is directed to shut down after it has completed processing all the changes propagated from the secondary server.

Example 3-33 shows the command that can be used to achieve this function. To determine whether the Q Apply program has stopped, you should monitor the contents of the Q Apply log, which shows the program as having stopped, as shown in Example 3-34.

Example 3-33 Autostop Q Apply on the primary server SC53

```
F QAPPITSO, CHGPARMS AUTOSTOP=Y
ASN0523I  "Q Apply" : "ITSO" : "Initial" : The CHGPARMS command
response: "AUTOSTOP" has been set to "Y".
-                                          --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QAPPITSO          QAPP        00  26411    .95    .00   59.4   671K
 0      0      0      0     0
IEF404I QAPPITSO - ENDED - TIME=18.31.09 - ASID=0031 - SC53
-QAPPITSO ENDED.  NAME-                     TOTAL CPU TIME=   .95
TOTAL ELAPSED TIME=  59.4
$HASP395 QAPPITSO ENDED
$HASP309 INIT A    INACTIVE ******** C=ABCDE
SE '18.31.09 JOB28810 $HASP165 QAPPITSO ENDED AT WTSCPLX1  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

Example 3-34 Autostop Q Apply log contents on the primary server SC53

```
......................
2006-02-27-18.30.59.553321 <handleCHGPARMS> ASN0523I  "Q Apply" : "ITSO" : "Initial" : The
CHGPARMS command response: "AUTOSTOP" has been set to "Y".
2006-02-27-18.31.00.184309 <brwzMain> ASN8999D  "Q Apply" : "ITSO" : "BR00000" : Browser
for queue 'QREP.POKB.TO.POKA.RECVQ' will be suspended because AUTOSTOP option was
specified.
......................
2006-02-27-18.31.04.192190 <browser::~browser> ASN8999D  "Q Apply" : "ITSO" : "BR00000" :
browser for queue 'QREP.POKB.TO.POKA.RECVQ' found max msgID
'51524550440376A4000000000000000000000000000007533' after cleaning IBMQREP_DONEMSG table on
shutdown.
```

```
.........................
2006-02-27-18.31.08.572344 <asnqapp::main> ASN0573I  "Q Apply" : "ITSO" : "Initial" : The
program was stopped.
```

STEP SWUFOPB8: Verify RECVQ on primary consumed

Verify that the receive queue on the primary server is fully consumed, as described in "STEP SWCF10: Verify RECVQ on primary consumed" on page 57.

STEP SWUFOPB9: Delete messages in relevant queues on both servers

As a precautionary measure, delete any messages that may exist in the transmit, receive, admin, and restart queues on both the primary and secondary servers. This ensures that the restoration of normal operations does not encounter unexpected problems due to residual messages in the various queues.

Example 3-35 shows the deletion of all the messages in the various queues on the primary server, while Example 3-35 on page 76 shows the deletion on the secondary server.

Example 3-35 Delete messages in XMITQ, RECVQ, ADMINQ, and RESTARTQ on primary server SC53

```
1 CSQU000I CSQUTIL IBM WebSphere MQ for z/OS V6
0 CSQU001I CSQUTIL Queue Manager Utility - 2006-02-27 18:34:06
   COMMAND DDNAME(CMDINP)
- CSQU127I  Executing COMMAND using input from CMDINP data set
0 CSQU120I  Connecting to MQZ1
  CSQU121I  Connected to queue manager MQZ1
0 CSQU055I  Target queue manager is MQZ1
 STOP CHANNEL(MQZ1.TO.MQ59)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
 CSQM134I =MQZ1 CSQMTCHL  STOP CHANNEL(MQZ1.TO.MQ59) COMMAND ACCEPTED
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I =MQZ1 CSQXCRPS ' STOP CHANNEL' NORMAL COMPLETION
0CLEAR QLOCAL(QREP.POKB.TO.POKA.RECVQ)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I =MQZ1 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0CLEAR QLOCAL(QREP.POKA.ADMINQ)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I =MQZ1 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0CLEAR QLOCAL(QREP.POKA.RESTARTQ)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I =MQZ1 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0CLEAR QLOCAL(MQ59XMIT)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I =MQZ1 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0DISPLAY QUEUE(QREP.POKB.TO.POKA.RECVQ) CURDEPTH
0CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I =MQZ1
  QUEUE(QREP.POKB.TO.POKA.RECVQ)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I =MQZ1 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0DISPLAY QUEUE(QREP.POKA.ADMINQ) CURDEPTH
0CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I =MQZ1
  QUEUE(QREP.POKA.ADMINQ)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I =MQZ1 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0DISPLAY QUEUE(QREP.POKA.RESTARTQ) CURDEPTH
1CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I =MQZ1
  QUEUE(QREP.POKA.RESTARTQ)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I =MQZ1 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0DISPLAY QUEUE(MQ59XMIT) CURDEPTH
0CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I =MQZ1
  QUEUE(MQ59XMIT)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I =MQZ1 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0START CHANNEL(MQZ1.TO.MQ59)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
 CSQM134I =MQZ1 CSQMSCHL  START CHANNEL(MQZ1.TO.MQ59) COMMAND ACCEPTED
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I =MQZ1 CSQXCRPS ' START CHANNEL' NORMAL COMPLETION
0 CSQU057I  10 commands read
0 CSQU058I  10 commands issued and responses received, 0 failed
- CSQU143I  1 COMMAND statements attempted
0 CSQU144I  1 COMMAND statements executed successfully
0 CSQU148I CSQUTIL Utility completed, return code=0
```

Example 3-36 Delete messages in XMITQ, RECVQ, ADMINQ, and RESTARTQ on secondary server SC59

```
1 CSQU000I CSQUTIL IBM WebSphere MQ for z/OS V6
0 CSQU001I CSQUTIL Queue Manager Utility - 2006-02-27 18:34:17
   COMMAND DDNAME(CMDINP)
- CSQU127I  Executing COMMAND using input from CMDINP data set
0 CSQU120I  Connecting to MQ59
  CSQU121I  Connected to queue manager MQ59
0 CSQU055I  Target queue manager is MQ59
 STOP CHANNEL(MQ59.TO.MQZ1)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
 CSQM134I -MQ59 CSQMTCHL  STOP CHANNEL(MQ59.TO.MQZ1) COMMAND ACCEPTED
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQ59 CSQXCRPS ' STOP CHANNEL' NORMAL COMPLETION
0CLEAR QLOCAL(QREP.POKA.TO.POKB.RECVQ)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQ59 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0CLEAR QLOCAL(QREP.POKB.ADMINQ)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQ59 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0CLEAR QLOCAL(QREP.POKB.RESTARTQ)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQ59 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0CLEAR QLOCAL(MQZ1XMIT)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQ59 CSQMRQLC ' CLEAR QLOCAL' NORMAL COMPLETION
0DISPLAY QUEUE(QREP.POKA.TO.POKB.RECVQ) CURDEPTH
0CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I -MQ59
  QUEUE(QREP.POKA.TO.POKB.RECVQ)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I -MQ59 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0DISPLAY QUEUE(QREP.POKB.ADMINQ) CURDEPTH
0CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I -MQ59
  QUEUE(QREP.POKB.ADMINQ)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I -MQ59 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0DISPLAY QUEUE(QREP.POKB.RESTARTQ) CURDEPTH
1CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I -MQ59
  QUEUE(QREP.POKB.RESTARTQ)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I -MQ59 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0DISPLAY QUEUE(MQZ1XMIT) CURDEPTH
0CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
 CSQM401I -MQ59
  QUEUE(MQZ1XMIT)
  TYPE(QLOCAL)
  QSGDISP(QMGR)
  CURDEPTH(0)
 CSQ9022I -MQ59 CSQMDRTS ' DISPLAY QUEUE' NORMAL COMPLETION
0START CHANNEL(MQ59.TO.MQZ1)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
 CSQM134I -MQ59 CSQMSCHL  START CHANNEL(MQ59.TO.MQZ1) COMMAND ACCEPTED
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQ59 CSQXCRPS ' START CHANNEL' NORMAL COMPLETION
0 CSQU057I  10 commands read
0 CSQU058I  10 commands issued and responses received, 0 failed
- CSQU143I  1 COMMAND statements attempted
0 CSQU144I  1 COMMAND statements executed successfully
0 CSQU148I CSQUTIL Utility completed, return code=0
```

STEP SWUFOPB10: Analyze and resolve all exceptions

Exceptions may be generated in the IBMQREP_EXCEPTIONS table at the primary server only. There are no exceptions at the secondary server because unpropagated changes from the primary server to the secondary server are ignored in this scenario. A careful analysis of these exceptions must be conducted and resolved to ensure that the data at the primary server is correct. For a detailed understanding of the causes of conflict and SQL error exceptions, especially those that cause data inconsistencies, refer to Appendix B, “Exception processing in a bidirectional Q replication environment” on page 191.

Once all the exceptions have been identified and resolved, the data on the secondary server must be reinitialized with the data on the primary server. This is achieved by first deactivating all the subscriptions, and then activating all the subscriptions using an automatic load (loadphase ‘I’) or a NO load (loadphase ‘N’).

- If the automatic load option is chosen, follow steps “STEP SWUFOPB10Ia: Resume primary workload” on page 79 and “STEP SWUFOPB10Ib: Deactivate subscriptions loadphase 'I'” on page 79 before executing “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 82.

- If the NO load option is chosen, follow steps “STEP SWUFOPB10Na: Resync database on secondary from primary” on page 81 and “STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'” on page 81 before executing “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 82.

STEP SWUFOPB10Ia: Resume primary workload

The application workload can now be started on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

Note: The application workload can be started now because Q Apply uses the spill queue to store propagated changes until the load phase is completed. It is therefore important to ensure that adequate spill queue space is provided to accommodate all the propagated changes that may occur until the load is completed.

STEP SWUFOPB10Ib: Deactivate subscriptions loadphase 'I'

For the automatic load option, subscriptions can be deactivated by setting:

- Columns STATE=’N’, SUB_ID=NULL, GROUP_MEMBERS=NULL, STATE_TRANSITION=NULL, and HAS_LOADPHASE=’I’ in the IBMQREP_SUBS table on the primary server. The value of ‘I’ in the HAS_LOADPHASE column indicates that an automatic load is specified.

  Note: Setting the STATE column to ‘N’ identifies the subscription as new, which causes the Q Capture program to automatically activate this Q subscription when the program is started or reinitialized.

- Columns STATE=’I’, SUB_ID=NULL, GROUP_MEMBERS=NULL, STATE_TRANSITION=NULL, and HAS_LOADPHASE=’I’ in the IBMQREP_SUBS table on the secondary server.

  Note: The STATE column in the IBMQREP_SUBS table on the secondary server is set to ‘I’ even though the corresponding value on the primary server is ‘N’.

- Columns STATE=’I’ and SUB_ID=’NULL’ in the IBMQREP_TARGETS table on the primary and secondary server.
- Column STATE=’A’ in the IBMQREP_RECVQUEUES table on the primary and secondary server. Setting this STATE column value to ‘A’ indicates that the queue is active and transactions are being retrieved from this queue by the worker thread.
- Column STATE=’A’ in the IBMQREP_SENDQUEUES table on the primary and secondary server. Setting this STATE column value to ‘A’ indicates that the queue is active and transactions are being written to this queue by the worker thread.

Attention: When STATE=’N’ in the IBMQREP_SUBS table on the primary server, and STATE=‘I’ in the IBMQREP_SUBS table on the secondary server, the table in the secondary server is refreshed from the data in the table on the primary server.

When STATE=’I’ in the IBMQREP_SUBS table on the primary server, and STATE=‘N’ in the IBMQREP_SUBS table on the secondary server, the table in the primary server is refreshed from the data in the table on the secondary server.

Example 3-37 shows the SQL for deactivating subscriptions on the primary server, while Example 3-38 shows the same for the secondary server.

Example 3-37 Deactivate subscriptions on the primary server SC53

```
D:\SG247215\Code>db2 -tvf update_state_ci_source.sql
CONNECT TO d8g1 user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = D8G1


UPDATE ITSO.IBMQREP_SUBS SET state='N', sub_id=NULL, group_members=NULL,
state_transition=NULL, has_loadphase = 'I'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_TARGETS SET state='I', sub_id=NULL
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_RECVQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_SENDQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

CONNECT RESET
DB20000I  The SQL command completed successfully.
```

Example 3-38 Deactivate subscriptions on the secondary server SC59

```
D:\SG247215\Code>db2 -tvf update_state_ci_target.sql
CONNECT TO db8a user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = DB8A


UPDATE ITSO.IBMQREP_SUBS SET state='I', sub_id=NULL, group_members=NULL,
state_transition=NULL, has_loadphase = 'I'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_TARGETS SET state='I', sub_id=NULL
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_RECVQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_SENDQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

CONNECT RESET
DB20000I  The SQL command completed successfully.
```

STEP SWUFOPB10Na: Resync database on secondary from primary

Since the NO load option is chosen, it is your responsibility to unload data from the primary server and restore it on the secondary server. We do not recommend any particular approach to achieve this function.

STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'

For the NO load option, subscriptions can be deactivated by setting:

- Columns STATE='I', SUB_ID=NULL, GROUP_MEMBERS=NULL, STATE_TRANSITION=NULL, and HAS_LOADPHASE='N' in the IBMQREP_SUBS table on the primary server. The value of 'N' in the HAS_LOADPHASE column indicates that a NO load is specified.
- Columns STATE='I' and SUB_ID='NULL' in the IBMQREP_TARGETS table on the primary and secondary server.
- Column STATE='A' in the IBMQREP_RECVQUEUES table on the primary and secondary server. Setting this STATE column value to 'A' indicates that the queue is active and transactions are being retrieved from this queue by the worker thread.
- Column STATE='A' in the IBMQREP_SENDQUEUES table on the primary and secondary server. Setting this STATE column value to 'A' indicates that the queue is active and transactions are being written to this queue by the worker thread.

Example 3-39 shows the SQL for deactivating subscriptions on the primary server, while Example 3-40 shows the same for the secondary server.

Example 3-39 Deactivate subscriptions on the primary server SC53

```
D:\SG247215\Code>db2 -tvf update_state_source.sql
CONNECT TO d8g1 user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = D8G1


UPDATE ITSO.IBMQREP_SUBS SET state='I', sub_id=NULL, group_members=NULL, state_t
ransition=NULL, has_loadphase = 'N'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_TARGETS SET state='I', sub_id=NULL
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_RECVQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_SENDQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

CONNECT RESET
DB20000I  The SQL command completed successfully.
```

Example 3-40 Deactivate subscriptions on the secondary server SC59

```
D:\SG247215\Code>db2 -tvf update_state_cn_target.sql
CONNECT TO db8a user mwiman using

   Database Connection Information
```

```
 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = DB8A


UPDATE ITSO.IBMQREP_SUBS SET state='I', sub_id=NULL, group_members=NULL,
state_transition=NULL, has_loadphase = 'N'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_TARGETS SET state='I', sub_id=NULL
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_RECVQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_SENDQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

CONNECT RESET
DB20000I  The SQL command completed successfully.
```

STEP SWUFOPB11: Cold start Q Capture on both servers

After the subscriptions have been deactivated on both the primary and secondary server, regardless of the automatic or NO load option, cold start Q Capture on both the primary and secondary server.

A Q Capture cold start replaces the restart message in the restart queue with a message that causes Q Capture to start processing log records at the current point in the log.

- Cold start Q Capture on the primary server by submitting the job shown in Example 3-41, the output of which indicates that the Q Capture program initialized successfully, as shown in Example 3-42. You can monitor the contents of the Q Capture log, which shows the Q Capture program initializing successfully, as shown in Example 3-43 on page 83.
- Cold start Q Capture on the secondary server by submitting the job shown in Example 3-44, the output of which indicates that the Q Capture program initialized successfully, as shown in Example 3-45 on page 84. You can monitor the contents of the Q Capture log, which shows the Q Capture program initializing successfully, as shown in Example 3-46 on page 84.

Example 3-41 JCL to cold start Q Capture on the primary server SC53

```
//QCAPITSO JOB NOTIFY=&SYSUID,                                    JOB17436
//         MSGCLASS=H,MSGLEVEL=(1,0),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//*****************************************************************
//QCAP     EXEC PGM=ASNQCAP,
// PARM='/CAPTURE_SERVER=D8G1 capture_schema=ITSO startmode=cold
//             CAPTURE_PATH=//''HUEYKIM'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//         DD DSN=MQ600.SCSQANLE,DISP=SHR
//         DD DSN=MQ600.SCSQLOAD,DISP=SHR
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,70)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
```

```
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

Example 3-42 Cold start Q Capture on the primary server SC53 - job submission output

```
$HASP100 QCAPITSO ON INTRDR                          FROM TSU28509
HUEYKIM
IRR010I  USERID HUEYKIM  IS ASSIGNED TO THIS JOB.
ICH70001I HUEYKIM  LAST ACCESS AT 18:34:07 ON MONDAY, FEBRUARY 27,
2006
$HASP373 QCAPITSO STARTED - INIT A    - CLASS A - SYS SC53
IEF403I QCAPITSO - STARTED - TIME=18.59.08 - ASID=0031 - SC53
$HASP100 BPXAS    ON STCINRDR
$HASP373 BPXAS    STARTED
IEF403I BPXAS - STARTED - TIME=18.59.10 - ASID=0062 - SC53
ASN7000I  "Q Capture" : "ITSO" : "WorkerThread" : "0" 584
subscriptions are active. "0" subscriptions are inactive. "0"
subscriptions that were new and were successfully activated. "0"
subscriptions that were new could not be activated and are now
inactive.
ASN0572I  "Q Capture" : "ITSO" : "WorkerThread" : The program
initialized successfully.
```

Example 3-43 Cold start Q Capture log contents on the primary server SC53

```
................................
2006-02-27-18.59.14.420222 <asnParmClass::printParms> ASN0529I  "Q Capture" : "ITSO" :
"Initial" : The value of "STARTMODE" was set to "COLD" at startup by the following method:
"COMMANDLINE".
..............................
2006-02-27-18.59.15.654830 <waitForLogrdInit> ASN7108I  "Q Capture" : "ITSO" :
"WorkerThread" :  At program initialization, the highest log sequence number of a
successfully processed transaction is "0000:0000:0000:0000" and the lowest log sequence
number of a transaction still to be committed is "BE6D:FC82:FB91:0000".
2006-02-27-18.59.15.659348 <asnqwk> ASN0572I  "Q Capture" : "ITSO" : "WorkerThread" : The
program initialized successfully.
```

Example 3-44 JCL to cold start Q Capture on the secondary server SC59

```
//QCAPITSO JOB (POK,999),HUEYKIM,MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M,TIME=NOLIMIT
//QCAP     EXEC PGM=ASNQCAP,
// PARM='/CAPTURE_SERVER=DB8A capture_schema=ITSO startmode=cold
//             CAPTURE_PATH=//''HUEYKIM'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8A8.SDSNLOAD,DISP=SHR
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,70)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
```

Example 3-45 Cold start Q Capture on the secondary server SC59 - job submission output

```
$HASP100 QCAPITSO ON INTRDR      HUEYKIM              FROM TSU13316
HUEYKIM
IRR010I  USERID HUEYKIM  IS ASSIGNED TO THIS JOB.
ICH70001I HUEYKIM  LAST ACCESS AT 18:34:17 ON MONDAY, FEBRUARY 27,
2006
$HASP373 QCAPITSO STARTED - INIT 1    - CLASS A - SYS IBM2
ASN7000I  "Q Capture" : "ITSO" : "WorkerThread" : "0" 927
subscriptions are active. "4" subscriptions are inactive. "0"
subscriptions that were new and were successfully activated. "0"
subscriptions that were new could not be activated and are now
inactive.
ASN0572I  "Q Capture" : "ITSO" : "WorkerThread" : The program
initialized successfully.
```

Example 3-46 Cold start Q Capture log contents on the secondary server SC59

```
.......................
2006-02-27-19.00.25.775755 <asnParmClass::printParms> ASN0529I  "Q Capture" : "ITSO" :
"Initial" : The value of "STARTMODE" was set to "COLD" at startup by the following method:
"COMMANDLINE".
..........................
2006-02-27-19.00.26.828275 <subMgr::processSubscriptions> ASN7000I  "Q Capture" : "ITSO" :
"WorkerThread" : "0" subscriptions are active. "4" subscriptions are inactive. "0"
subscriptions that were new and were successfully activated. "0" subscriptions that were
new could not be activated and are now inactive.
......................
2006-02-27-19.00.26.842981 <waitForLogrdInit> ASN7108I  "Q Capture" : "ITSO" :
"WorkerThread" :  At program initialization, the highest log sequence number of a
successfully processed transaction is "0000:0000:0000:0000" and the lowest log sequence
number of a transaction still to be committed is "0000:0FFC:56AC:0000".
2006-02-27-19.00.26.843046 <asnqwk> ASN0572I  "Q Capture" : "ITSO" : "WorkerThread" : The
program initialized successfully.
```

STEP SWUFOPB12: Start Q Apply on both servers

Start Q Apply on the primary server as described in "STEP SWCF1: Start Q Apply on primary" on page 47, and on the secondary server as described in "STEP SWCF4: Start Q Apply on secondary" on page 51.

STEP SWUFOPB13: Verify Q replication up and running on both servers

Verify that normal operations have been restored as described in "STEP SWUFOPA17: Verify that Q replication up and running" on page 70.

STEP SWUFOPB14: Resume RO workload on secondary

The read-only workload can now be resumed on the secondary server.

STEP SWUFOPB15: If loadphase 'N', resume primary workload

If the NO load option had been chosen, start the application workload on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

Note: For the automatic load option, the application workload was started in "STEP SWUFOPB10Ia: Resume primary workload" on page 79.

3.7.4 Option C: Resync secondary and reinitialize primary

Table 3-1 on page 37 summarizes the considerations when choosing between four possible switchback options. Option C involves warm starting Q Capture on the primary server when it becomes available, but not Q Apply on the primary server right away. This option involves re-synchronizing the secondary server with unpropagated changes from the primary server. Once that is completed, the subscriptions are deactivated and reactivated so that the primary server is re-initialized with the contents of the secondary server.

With an uncontrolled failover, the procedure for switching back to the primary server and establishing normal operations is far more complex than that of a switchback after a controlled failover.

The option C switchback procedure after a successful uncontrolled failover is shown in Figure 3-29 and assumes the following:

- The primary server is up and running DB2.
- WebSphere MQ is up and running on both the primary and secondary servers.
- The primary server is not running any workload—read-only or update.
- Q Capture and Q Apply are not running on either the primary or secondary servers.
- All the subscriptions are in an active state.
- The application workload is running on the secondary server, causing changes to be written to the DB2 log.
- Switchback is performed during a maintenance period when the throughput is low.
- Potential conflict and SQL error exceptions are recorded in the IBMQREP_EXCEPTIONS table on the secondary server. There are no exceptions at the primary server because unpropagated changes from the secondary server to the primary server are ignored in this scenario. These exceptions are likely to occur during switchback because of unpropagated changes from the primary server to the secondary server.

Attention: The guiding principle of this switchback procedure is to:

- Minimize the potential for exceptions and contention on the secondary server.
- Narrow the window of application workload unavailability by stopping the workload as late as possible and restarting it as early as possible.

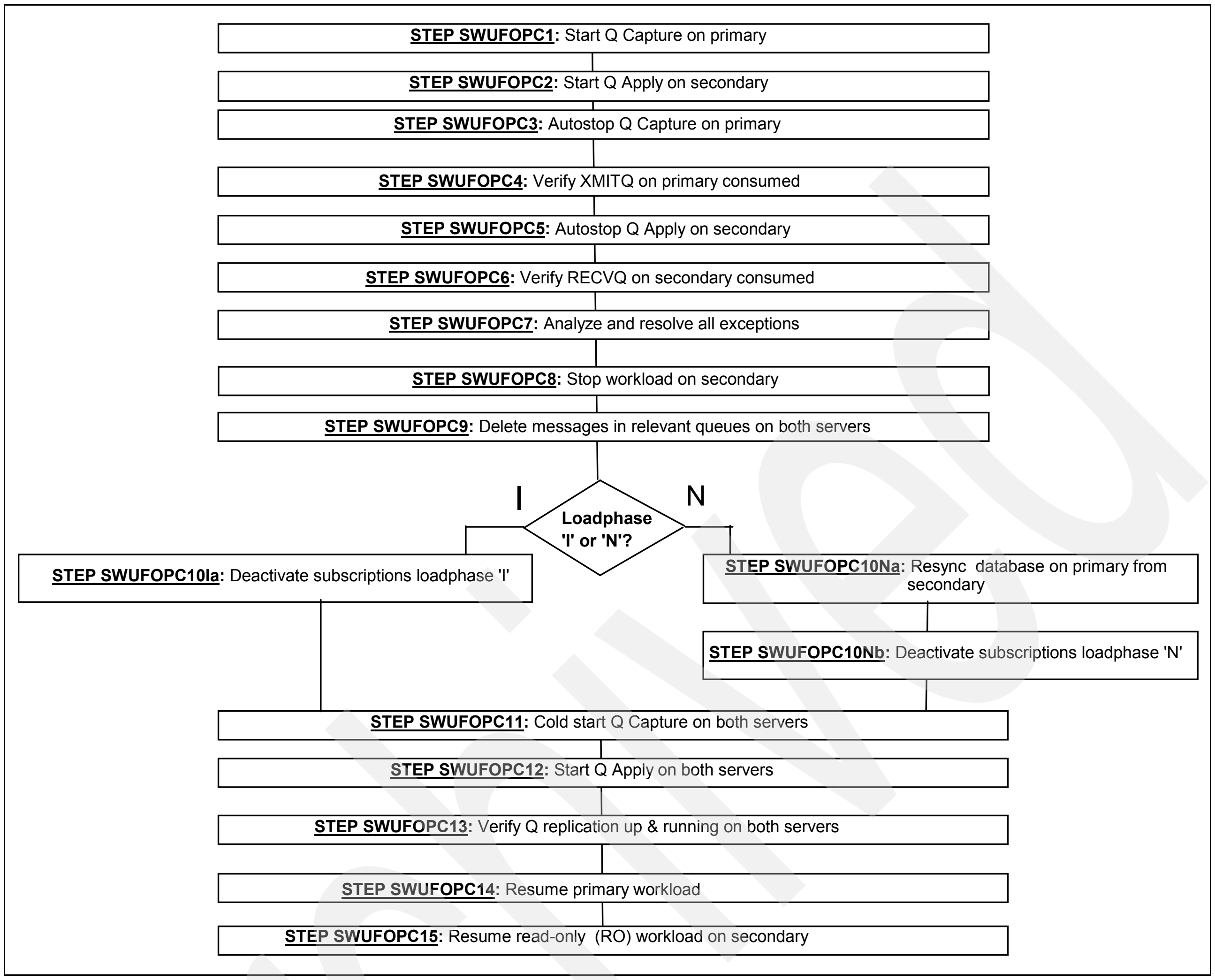

STEP SWUFOPC1: Start Q Capture on primary
STEP SWUFOPC2: Start Q Apply on secondary
STEP SWUFOPC3: Autostop Q Capture on primary
STEP SWUFOPC4: Verify XMITQ on primary consumed
STEP SWUFOPC5: Autostop Q Apply on secondary
STEP SWUFOPC6: Verify RECVQ on secondary consumed
STEP SWUFOPC7: Analyze and resolve all exceptions
STEP SWUFOPC8: Stop workload on secondary
STEP SWUFOPC9: Delete messages in relevant queues on both servers
I
N
Loadphase 'I' or 'N'?
STEP SWUFOPC10Ia: Deactivate subscriptions loadphase 'I'
STEP SWUFOPC10Na: Resync database on primary from secondary
STEP SWUFOPC10Nb: Deactivate subscriptions loadphase 'N'
STEP SWUFOPC11: Cold start Q Capture on both servers
STEP SWUFOPC12: Start Q Apply on both servers
STEP SWUFOPC13: Verify Q replication up & running on both servers
STEP SWUFOPC14: Resume primary workload
STEP SWUFOPC15: Resume read-only (RO) workload on secondary

Figure 3-29 Overview of Option C steps

Each of the steps shown in Figure 3-29 is described briefly in the following sections.

STEP SWUFOPC1: Start Q Capture on primary

Start Q Capture on the primary server, as described in “STEP SWCF2: Start Q Capture on primary” on page 49.

STEP SWUFOPC2: Start Q Apply on secondary

Start Q Apply on the primary server, as described in “STEP SWCF4: Start Q Apply on secondary” on page 51.

STEP SWUFOPC3: Autostop Q Capture on primary

In this step, the Q Capture program on the primary server is directed to shut down after it has completed processing all the committed changes that have been captured while the workload was running on the primary server, and written them to the send queue, as described in “STEP CF2: Autostop Q Capture on primary” on page 41.

STEP SWUFOPC4: Verify XMITQ on primary consumed

Verify that the transmit queue on the primary server is fully consumed, as described in "STEP CF5: Verify XMITQ on primary consumed" on page 44.

STEP SWUFOPC5: Autostop Q Apply on secondary

In this step, the Q Apply program on the secondary server is directed to shut down after it has completed processing all the changes propagated from the primary server, as described in "STEP CF7: Autostop Q Apply on secondary" on page 45.

STEP SWUFOPC6: Verify RECVQ on secondary consumed

Verify that the receive queue on the secondary server is fully consumed, as described in "STEP UF1a: Verify RECVQ on secondary consumed" on page 64.

STEP SWUFOPC7: Analyze and resolve all exceptions

Exceptions may be generated in the IBMQREP_EXCEPTIONS table at the secondary server only. There are no exceptions at the primary server because unpropagated changes from the secondary server to the primary server are ignored in this scenario. A careful analysis of these exceptions must be conducted and resolved to ensure that the data at the secondary server is correct. For a detailed understanding of the causes of conflict and SQL error exceptions, especially those that cause data inconsistencies, refer to Appendix B, "Exception processing in a bidirectional Q replication environment" on page 191.

Once all the exceptions have been identified and resolved, the data on the primary server must be reinitialized with the data on the secondary server.

STEP SWUFOPC8: Stop workload on secondary

In this step, stop the application workload directed to this server without redirecting it to the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP SWUFOPC9: Delete messages in relevant queues on both servers

As a precautionary measure, delete any messages that may exist in the transmit, receive, admin, and restart queues on both the primary and secondary servers, as described in "STEP SWUFOPB9: Delete messages in relevant queues on both servers" on page 76.

This ensures that the restoration of normal operations does not encounter unexpected problems due to residual messages in the various queues.

STEP SWUFOPC10: Deactivate subscriptions

Once all the exceptions have been identified and resolved, the data on the primary server must be reinitialized with the data on the secondary server. This is achieved by first deactivating all the subscriptions and then activating all the subscriptions using an automatic load (loadphase 'I') or a NO load (loadphase 'N').

- If the automatic load option is chosen, follow "STEP SWUFOPC10Ia: Deactivate subscriptions loadphase 'I'" on page 88, followed by "STEP SWUFOPC11: Cold start Q Capture on both servers" on page 89.
- If the NO load option is chosen, follow "STEP SWUFOPB10Na: Resync database on secondary from primary" on page 81 and "STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'" on page 81 before executing "STEP SWUFOPB11: Cold start Q Capture on both servers" on page 82.

STEP SWUFOPC10Ia: Deactivate subscriptions loadphase ‘I’

For the automatic load option, subscriptions can be deactivated by setting:

- Columns STATE=’I’, SUB_ID=NULL, GROUP_MEMBERS=NULL, STATE_TRANSITION=NULL, and HAS_LOADPHASE=’I’ in the IBMQREP_SUBS table on the primary server. The value of ‘I’ in the HAS_LOADPHASE column indicates that an automatic load is specified.

  Note: The STATE column in the IBMQREP_SUBS table on the secondary server is set to ‘I’ even though the corresponding value on the secondary server is ‘N’.

- Columns STATE=’N’, SUB_ID=NULL, GROUP_MEMBERS=NULL, STATE_TRANSITION=NULL, and HAS_LOADPHASE=’I’ in the IBMQREP_SUBS table on the secondary server.

  Note: Setting the STATE column to ‘N’ identifies the subscription as new, which causes the Q Capture program to automatically activate this Q subscription when the program is started or reinitialized.

- Columns STATE=’I’ and SUB_ID=’NULL’ in the IBMQREP_TARGETS table on the primary and secondary server.
- Column STATE=’A’ in the IBMQREP_RECVQUEUES table on the primary and secondary server. Setting this STATE column value to ‘A’ indicates that the queue is active and transactions are being retrieved from this queue by the worker thread.
- Column STATE=’A’ in the IBMQREP_SENDQUEUES table on the primary and secondary server. Setting this STATE column value to ‘A’ indicates that the queue is active and transactions are being written to this queue by the worker thread.

Attention: When STATE=’N’ in the IBMQREP_SUBS table on the primary server, and STATE=‘I’ in the IBMQREP_SUBS table on the secondary server, the table in the secondary server is refreshed from the data in the table on the primary server.

When STATE=’I’ in the IBMQREP_SUBS table on the primary server, and STATE=‘N’ in the IBMQREP_SUBS table on the secondary server, the table in the primary server is refreshed from the data in the table on the secondary server.

Example 3-47 shows the SQL for deactivating subscriptions on the primary server, while Example 3-48 shows the same for the secondary server.

Example 3-47 Deactivate subscriptions on the primary server SC53

```
D:\SG247215\Code>db2 -tvf update_state_di_source.sql
CONNECT TO d8g1 user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = D8G1


UPDATE ITSO.IBMQREP_SUBS SET state='I', sub_id=NULL, group_members=NULL,
state_transition=NULL, has_loadphase = 'I'
DB20000I  The SQL command completed successfully.
```

```
UPDATE ITSO.IBMQREP_TARGETS SET state='I', sub_id=NULL
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_RECVQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_SENDQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

CONNECT RESET
DB20000I  The SQL command completed successfully.
```

Example 3-48   Deactivate subscriptions on the secondary server SC59

```
D:\SG247215\Code>db2 -tvf update_state_di_target.sql
CONNECT TO db8a user mwiman using

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5
 SQL authorization ID   = MWIMAN
 Local database alias   = DB8A


UPDATE ITSO.IBMQREP_SUBS SET state='N', sub_id=NULL, group_members=NULL,
state_transition=NULL, has_loadphase = 'I'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_TARGETS SET state='I', sub_id=NULL
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_RECVQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

UPDATE ITSO.IBMQREP_SENDQUEUES SET state='A'
DB20000I  The SQL command completed successfully.

CONNECT RESET
DB20000I  The SQL command completed successfully.
```

STEP SWUFOPC10Na: Resync database on primary from secondary

Since the NO load option is chosen, it is your responsibility to unload data from the secondary server and restore it on the primary server. We do not recommend any particular approach to achieve this function.

STEP SWUFOPC10Nb: Deactivate subscriptions loadphase ‘N’

For the NO load option, subscriptions can be deactivated, as described in “STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'” on page 81.

STEP SWUFOPC11: Cold start Q Capture on both servers

Cold start Q Capture on the primary and secondary servers, as described in “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 82.

STEP SWUFOPC12: Start Q Apply on both servers

Start Q Apply on the primary server, as described in “STEP SWCF1: Start Q Apply on primary” on page 47, and on the secondary server, as described in “STEP SWCF4: Start Q Apply on secondary” on page 51.

STEP SWUFOPC13: Verify Q replication up and running on both servers

Verify that Q replication is up and running on both servers, as described in "STEP SWUFOPA17: Verify that Q replication up and running" on page 70.

When automatic load is chosen for a subscription, the Q Apply program on the primary server performs the load from the secondary server when the subscription is activated by the cold start of Q Capture on the secondary server since the subscription's state identifies it as a new subscription. The STATE column in the IBMQREP_SUBS table had been updated to 'N' on the secondary server in "STEP SWUFOPC10Ia: Deactivate subscriptions loadphase 'I'" on page 88.

The automatic load statistics of the four activated subscriptions shown in Example 3-49 through Example 3-52 on page 94 are obtained from the load trace files (one per subscription) such as HUEYKIM.DB8A.ITSO.S0000003.LOADTRC.

Example 3-49 Automatic load output 1/4

```
The current timestamp is 2006-02-27-22.13.25.939612

ASNQLOAD(compiled at 11:40:03 on Jan 19 2006) input values:
   Target Server  = DB8G
   Target Owner   = ITSO
   Target Table.  = TRAN_V1
   Source Server  = DB8A
   SQL.Stmt       = SELECT "ACCT_ID", "TRAN_ID", "TRAN_DATE", "TRAN_TYPE", "TRAN_QTY" FROM
"ITSO"."TRAN_V1"
Connect to TRG DB8G
--> INDEXES  == 1
--> FOR_KEYS == 0
--> getSourceTable() = "ITSO"."TRAN_V1"
Connect to SRC DB8A
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "ITSO"."TRAN_V1"
--> on server = DB8A
--> RET = 0
--> num_rows   == 0
--> getSortKeysTotal() == 0

--------> Running LOAD...

ASNLOAD Running under user = HUEYKIM
sIdBuf is 00000001
uIdBuf is ITSO00000001
WorkDSN1 = HUEYKIM.S0000001.UNLDEI
WorkDSN2 = HUEYKIM.S0000001.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000001.UNLDE MAXPRIME 4300 TEMPLATE WORKDSN1
DSN HUEYKIM.D8G1.ITSO.S0000001.UNLDEI  SP
ACE (50,100) CYL MAXPRIME 4300 TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000001.UNLDEO
SPACE (50,100) CYL MAXPRIME 4300 EXEC SQL
DECLARE C1 CURSOR FOR SELECT "ACCT_ID", "TRAN_ID", "TRAN_DATE", "TRAN_TYPE", "TRAN_QTY"
FROM "DB8A"."ITSO"."TRAN_V1"  ENDEXEC LOAD
 DATA INCURSOR(C1)  LOG NO NOCOPYPEND  WORKDDN(WORKDSN1,WORKDSN2)  REPLACE SORTKEYS 0 INTO
TABLE "ITSO"."TRAN_V1" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ITSO00000001
 DSNU1044I   DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000001.UNLDE MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
```

```
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN HUEYKIM.D8G1.ITSO.S0000001.UNLDEI SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000001.UNLDEO SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "ACCT_ID", "TRAN_ID",
"TRAN_DATE", "TRAN_TYPE",
 "TRAN_QTY" FROM "DB8A"."ITSO"."TRAN_V1" ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) LOG NO NOCOPYPEND WORKDDN(WORKDSN1,
WORKDSN2) REPLACE SORTKEYS 0
 DSNU650I  -D8G1 DSNURWI -  INTO TABLE "ITSO"."TRAN_V1" STATISTICS
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN1
                        DDNAME=SYS00001
                        DSN=HUEYKIM.D8G1.ITSO.S0000001.UNLDEI
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN2
                        DDNAME=SYS00002
                        DSN=HUEYKIM.D8G1.ITSO.S0000001.UNLDEO
 DSNU350I  -D8G1 DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU304I  -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE
ITSO.TRAN_V1
 DSNU1147I -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - TOTAL NUMBER OF RECORDS LOADED=0 FOR
TABLESPACE DSNDB04.TRANRV1
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=0
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:01
 DSNU610I  -D8G1 DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR DSNDB04.TRANRV1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR ITSO.TRAN_V1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DSNDB04.TRANRV1 SUCCESSFUL
 DSNU620I  -D8G1 DSNUSEF2 - RUNSTATS CATALOG TIMESTAMP = 2006-02-27-22.13.28.889694
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
--------> LOAD ended with CODE = 0

Exiting ASNLOAD
```

Example 3-50   Automatic load output 2/4

```
The current timestamp is 2006-02-27-22.13.26.552826

ASNQLOAD(compiled at 11:40:03 on Jan 19 2006) input values:
   Target Server  = DB8G
   Target Owner   = ITSO
   Target Table.  = PROD_V1
   Source Server  = DB8A
   SQL.Stmt       = SELECT "ACCT_ID", "PROD_DATE", "PROD_TYPE", "PROD_QTY" FROM
"ITSO"."PROD_V1"
Connect to TRG DB8G
--> INDEXES  == 1
--> FOR_KEYS == 0
--> getSourceTable() = "ITSO"."PROD_V1"
Connect to SRC DB8A
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "ITSO"."PROD_V1"
--> on server = DB8A
--> RET = 0
--> num_rows   == 0
--> getSortKeysTotal() == 0

--------> Running LOAD...

ASNLOAD Running under user = HUEYKIM
```

```
sIdBuf is 00000002
uIdBuf is ITSO00000002
WorkDSN1 = HUEYKIM.S0000002.UNLDEI
WorkDSN2 = HUEYKIM.S0000002.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000002.UNLDE MAXPRIME 4300 TEMPLATE WORKDSN1
DSN HUEYKIM.D8G1.ITSO.S0000002.UNLDEI  SP
ACE (50,100) CYL MAXPRIME 4300 TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000002.UNLDEO
SPACE (50,100) CYL MAXPRIME 4300 EXEC SQL
DECLARE C1 CURSOR FOR SELECT "ACCT_ID", "PROD_DATE", "PROD_TYPE", "PROD_QTY" FROM
"DB8A"."ITSO"."PROD_V1"  ENDEXEC LOAD DATA INCUR
SOR(C1)  LOG NO NOCOPYPEND  WORKDDN(WORKDSN1,WORKDSN2)  REPLACE SORTKEYS 0 INTO TABLE
"ITSO"."PROD_V1" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ITSO00000002
 DSNU1044I   DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000002.UNLDE MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN HUEYKIM.D8G1.ITSO.S0000002.UNLDEI SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000002.UNLDEO SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "ACCT_ID", "PROD_DATE",
"PROD_TYPE", "PROD_QTY" FROM
 "DB8A"."ITSO"."PROD_V1" ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) LOG NO NOCOPYPEND WORKDDN(WORKDSN1,
WORKDSN2) REPLACE SORTKEYS 0
 DSNU650I  -D8G1 DSNURWI -  INTO TABLE "ITSO"."PROD_V1" STATISTICS
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN1
                        DDNAME=SYS00007
                        DSN=HUEYKIM.D8G1.ITSO.S0000002.UNLDEI
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN2
                        DDNAME=SYS00008
                        DSN=HUEYKIM.D8G1.ITSO.S0000002.UNLDEO
 DSNU350I  -D8G1 DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU304I  -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE
ITSO.PROD_V1
 DSNU1147I -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - TOTAL NUMBER OF RECORDS LOADED=0 FOR
TABLESPACE DSNDB04.PRODRV1
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=0
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:01
 DSNU610I  -D8G1 DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR DSNDB04.PRODRV1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR ITSO.PROD_V1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DSNDB04.PRODRV1 SUCCESSFUL
 DSNU620I  -D8G1 DSNUSEF2 - RUNSTATS CATALOG TIMESTAMP = 2006-02-27-22.13.42.102964
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
--------> LOAD ended with CODE = 0

Exiting ASNLOAD
```

Example 3-51 Automatic load output 3/4

```
The current timestamp is 2006-02-27-22.13.26.327535

ASNQLOAD(compiled at 11:40:03 on Jan 19 2006) input values:
   Target Server  = DB8G
   Target Owner   = ITSO
   Target Table.  = OPTIONS_V1
   Source Server  = DB8A
   SQL.Stmt       = SELECT "OPTION_ID", "NOTES", "UPDATED" FROM "ITSO"."OPTIONS_V1"
Connect to TRG DB8G
--> INDEXES  == 1
--> FOR_KEYS == 0
--> getSourceTable() = "ITSO"."OPTIONS_V1"
Connect to SRC DB8A
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "ITSO"."OPTIONS_V1"
--> on server = DB8A
--> RET = 0
--> num_rows   == 9
--> getSortKeysTotal() == 9

--------> Running LOAD...

ASNLOAD Running under user = HUEYKIM
sIdBuf is 00000003
uIdBuf is ITSO00000003
WorkDSN1 = HUEYKIM.S0000003.UNLDEI
WorkDSN2 = HUEYKIM.S0000003.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000003.UNLDE MAXPRIME 4300 TEMPLATE WORKDSN1
DSN HUEYKIM.D8G1.ITSO.S0000003.UNLDEI  SP
ACE (50,100) CYL MAXPRIME 4300 TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000003.UNLDEO
SPACE (50,100) CYL MAXPRIME 4300 EXEC SQL
DECLARE C1 CURSOR FOR SELECT "OPTION_ID", "NOTES", "UPDATED" FROM
"DB8A"."ITSO"."OPTIONS_V1"  ENDEXEC LOAD DATA INCURSOR(C1)  LOG
NO NOCOPYPEND  WORKDDN(WORKDSN1,WORKDSN2)  REPLACE SORTKEYS 9 INTO TABLE
"ITSO"."OPTIONS_V1" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ITSO00000003
 DSNU1044I   DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000003.UNLDE MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN HUEYKIM.D8G1.ITSO.S0000003.UNLDEI SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000003.UNLDEO SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "OPTION_ID", "NOTES",
"UPDATED" FROM
 "DB8A"."ITSO"."OPTIONS_V1" ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) LOG NO NOCOPYPEND WORKDDN(WORKDSN1,
WORKDSN2) REPLACE SORTKEYS 9
 DSNU650I  -D8G1 DSNURWI -  INTO TABLE "ITSO"."OPTIONS_V1" STATISTICS
 DSNU068I  -D8G1 DSNURWI - KEYWORD INLINE STATISTICS IS NOT SUPPORTED FOR LOB TABLE SPACES
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN1
                        DDNAME=SYS00003
                        DSN=HUEYKIM.D8G1.ITSO.S0000003.UNLDEI
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN2
                        DDNAME=SYS00004
                        DSN=HUEYKIM.D8G1.ITSO.S0000003.UNLDEO
 DSNU350I  -D8G1 DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU304I  -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=9 FOR TABLE
ITSO.OPTIONS_V1
```

```
 DSNU1147I -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - TOTAL NUMBER OF RECORDS LOADED=9 FOR
TABLESPACE ITSITDB1.ITSOLOBT
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=9
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:04
 DSNU042I    DSNUGSOR - SORT PHASE STATISTICS -
                       NUMBER OF RECORDS=9
                       ELAPSED TIME=00:00:00
 DSNU349I  -D8G1 DSNURBXA - BUILD PHASE STATISTICS - NUMBER OF KEYS=9 FOR INDEX
ITSO.OPTIONS_V1_PK
 DSNU258I    DSNURBXD - BUILD PHASE STATISTICS - NUMBER OF INDEXES=1
 DSNU259I    DSNURBXD - BUILD PHASE COMPLETE, ELAPSED TIME=00:00:00
 DSNU610I  -D8G1 DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR ITSITDB1.ITSOLOBT SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR ITSO.OPTIONS_V1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR ITSITDB1.ITSOLOBT SUCCESSFUL
 DSNU620I  -D8G1 DSNUSEF2 - RUNSTATS CATALOG TIMESTAMP = 2006-02-27-22.13.33.953846
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
--------> LOAD ended with CODE = 4

Exiting ASNLOAD
```

Example 3-52 Automatic load output 4/4

```
The current timestamp is 2006-02-27-22.13.26.458609

ASNQLOAD(compiled at 11:40:03 on Jan 19 2006) input values:
   Target Server  = DB8G
   Target Owner   = ITSO
   Target Table.  = BAL_V1
   Source Server  = DB8A
   SQL.Stmt       = SELECT "ACCT_ID", "ACCTG_RULE_CD", "BAL_TYPE_CD", "BAL_DATE",
"BAL_AMOUNT" FROM "ITSO"."BAL_V1"
Connect to TRG DB8G
--> INDEXES  == 2
--> FOR_KEYS == 0
--> getSourceTable() = "ITSO"."BAL_V1"
Connect to SRC DB8A
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "ITSO"."BAL_V1"
--> on server = DB8A
--> RET = 0
--> num_rows   == 57
--> getSortKeysTotal() == 114

--------> Running LOAD...

ASNLOAD Running under user = HUEYKIM
sIdBuf is 00000004
uIdBuf is ITSO00000004
WorkDSN1 = HUEYKIM.S0000004.UNLDEI
WorkDSN2 = HUEYKIM.S0000004.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000004.UNLDE MAXPRIME 4300 TEMPLATE WORKDSN1
DSN HUEYKIM.D8G1.ITSO.S0000004.UNLDEI  SP
ACE (50,100) CYL MAXPRIME 4300 TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000004.UNLDEO
SPACE (50,100) CYL MAXPRIME 4300 EXEC SQL
DECLARE C1 CURSOR FOR SELECT "ACCT_ID", "ACCTG_RULE_CD", "BAL_TYPE_CD", "BAL_DATE",
"BAL_AMOUNT" FROM "DB8A"."ITSO"."BAL_V1"  ENDE
XEC LOAD DATA INCURSOR(C1)  LOG NO NOCOPYPEND  WORKDDN(WORKDSN1,WORKDSN2)  REPLACE SORTKEYS
114 INTO TABLE "ITSO"."BAL_V1" STATIST
```

```
ICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ITSO00000004
 DSNU1044I   DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN HUEYKIM.D8G1.ITSO.S0000004.UNLDE MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN HUEYKIM.D8G1.ITSO.S0000004.UNLDEI SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN HUEYKIM.D8G1.ITSO.S0000004.UNLDEO SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "ACCT_ID", "ACCTG_RULE_CD",
"BAL_TYPE_CD",
 "BAL_DATE", "BAL_AMOUNT" FROM "DB8A"."ITSO"."BAL_V1" ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) LOG NO NOCOPYPEND WORKDDN(WORKDSN1,
WORKDSN2) REPLACE SORTKEYS 114
 DSNU650I  -D8G1 DSNURWI -  INTO TABLE "ITSO"."BAL_V1" STATISTICS
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN1
                        DDNAME=SYS00005
                        DSN=HUEYKIM.D8G1.ITSO.S0000004.UNLDEI
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN2
                        DDNAME=SYS00006
                        DSN=HUEYKIM.D8G1.ITSO.S0000004.UNLDEO
 DSNU350I  -D8G1 DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU395I    DSNURPIB - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 6
 DSNU304I  -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=57 FOR TABLE
ITSO.BAL_V1
 DSNU1147I -D8G1 DSNURWT - (RE)LOAD PHASE STATISTICS - TOTAL NUMBER OF RECORDS LOADED=57
FOR TABLESPACE DSNDB04.BALRV1
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=57
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:02
 DSNU394I  -D8G1 DSNURBXA - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=57 FOR INDEX
ITSO.BAL_V1_PK
 DSNU394I  -D8G1 DSNURBXA - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=57 FOR INDEX
ITSO.BAL_V1_UNQ1
 DSNU391I    DSNURPTB - SORTBLD PHASE STATISTICS. NUMBER OF INDEXES = 2
 DSNU392I    DSNURPTB - SORTBLD PHASE COMPLETE, ELAPSED TIME = 00:00:00
 DSNU610I  -D8G1 DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR DSNDB04.BALRV1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR ITSO.BAL_V1 SUCCESSFUL
 DSNU610I  -D8G1 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DSNDB04.BALRV1 SUCCESSFUL
 DSNU620I  -D8G1 DSNUSEF2 - RUNSTATS CATALOG TIMESTAMP = 2006-02-27-22.13.38.520234
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
--------> LOAD ended with CODE = 0

Exiting ASNLOAD
```

STEP SWUFOPC14: Resume primary workload

Start the application workload on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

STEP SWUFOPC15: Resume RO workload on secondary

The read-only workload can now be resumed on the secondary server.

3.7.5 Option D: Discard primary changes and reinitialize primary

Table 3-1 on page 37 summarizes the considerations when choosing between four possible switchback options. Option D involves discarding all unpropagated changes from the primary server, and re-initializing the primary server with the contents of the secondary server.

Attention: This is almost like a disaster recovery scenario, except that the primary site becomes available at a later point in time. In a disaster recovery scenario, all remnants of Q replication objects are first removed from the secondary server (disaster recovery site) and a new bidirectional replication configuration must be defined where this disaster recovery site becomes the primary server and another server becomes the new secondary server. “Remove all remnants of Q replication objects on a server” on page 100 describes the steps for removing all remnants of Q replication objects on the disaster recovery site.

With an uncontrolled failover, the procedure for switching back to the primary server and establishing normal operations is far more complex than that of a switchback after a controlled failover.

The option D switchback procedure after a successful uncontrolled failover is shown in Figure 3-30 and assumes the following:

- The primary server is up and running DB2.
- WebSphere MQ is up and running on both the primary and secondary servers.
- The primary server is not running any workload—read-only or update.
- Q Capture and Q Apply are not running on either the primary or secondary servers.
- All the subscriptions are in an active state.
- The application workload is running on the secondary server causing changes to be written to the DB2 log.
- Switchback is performed during a maintenance period when the throughput is low.
- There are no exceptions recorded in the IBMQREP_EXCEPTIONS table on either the primary or secondary server because all unpropagated changes are discarded.

Attention: The guiding principle of this switchback procedure is to narrow the window of application workload unavailability by stopping the workload as late as possible and restarting it as early as possible.

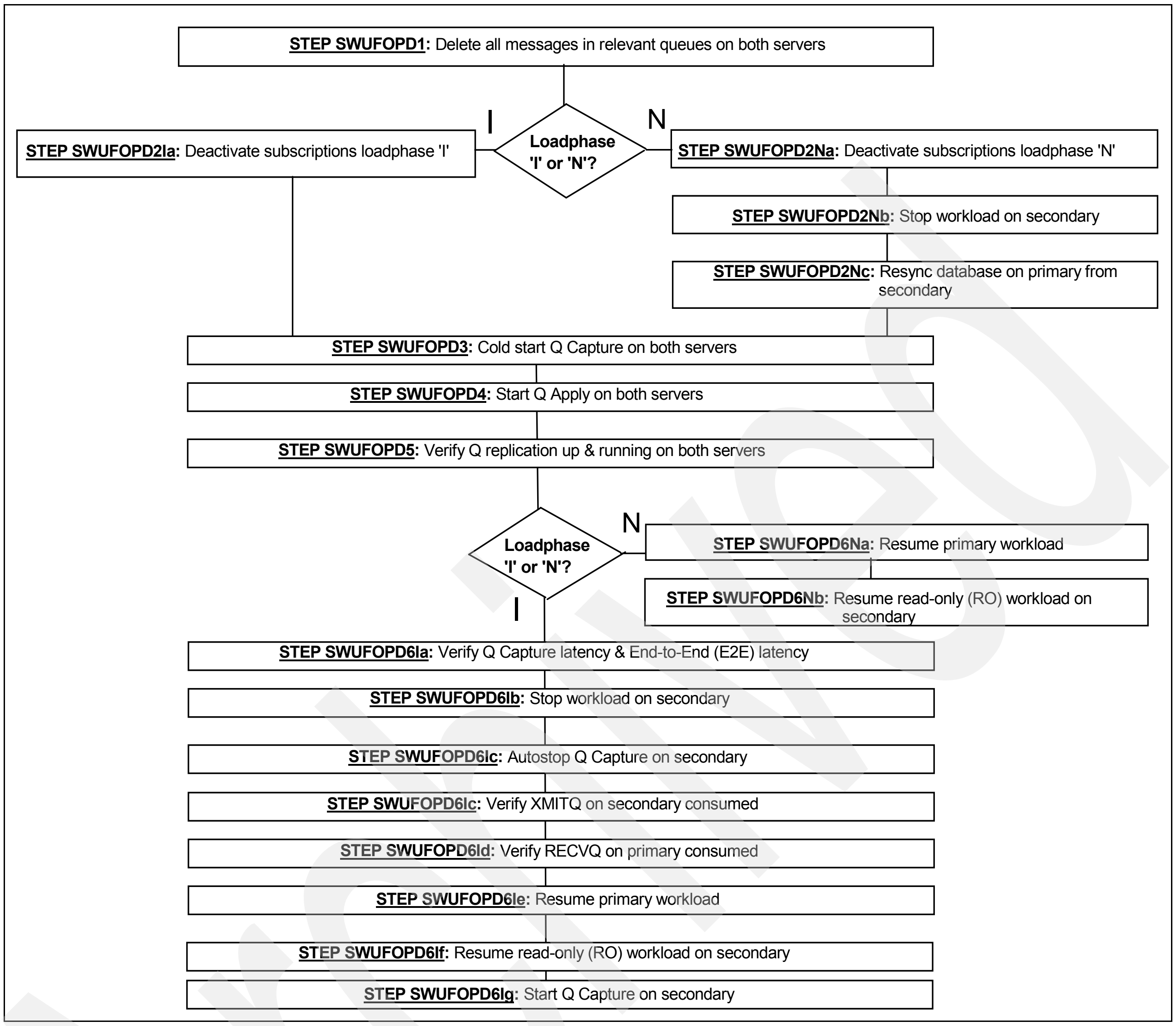

STEP SWUFOPD1: Delete all messages in relevant queues on both servers
Loadphase 'I' or 'N'?
I
N
STEP SWUFOPD2Ia: Deactivate subscriptions loadphase 'I'
STEP SWUFOPD2Na: Deactivate subscriptions loadphase 'N'
STEP SWUFOPD2Nb: Stop workload on secondary
STEP SWUFOPD2Nc: Resync database on primary from secondary
STEP SWUFOPD3: Cold start Q Capture on both servers
STEP SWUFOPD4: Start Q Apply on both servers
STEP SWUFOPD5: Verify Q replication up & running on both servers
Loadphase 'I' or 'N'?
N
I
STEP SWUFOPD6Na: Resume primary workload
STEP SWUFOPD6Nb: Resume read-only (RO) workload on secondary
STEP SWUFOPD6Ia: Verify Q Capture latency & End-to-End (E2E) latency
STEP SWUFOPD6Ib: Stop workload on secondary
STEP SWUFOPD6Ic: Autostop Q Capture on secondary
STEP SWUFOPD6Ic: Verify XMITQ on secondary consumed
STEP SWUFOPD6Id: Verify RECVQ on primary consumed
STEP SWUFOPD6Ie: Resume primary workload
STEP SWUFOPD6If: Resume read-only (RO) workload on secondary
STEP SWUFOPD6Ig: Start Q Capture on secondary

Figure 3-30 Overview of Option D steps

Each of the steps shown in Figure 3-30 is described briefly in the following sections.

STEP SWUFOPD1: Delete messages in relevant queues on both servers

As a precautionary measure, delete any messages that may exist in the transmit, receive, admin, and restart queues on both the primary and secondary servers, as described in "STEP SWUFOPB9: Delete messages in relevant queues on both servers" on page 76. This ensures that the restoration of normal operations does not encounter unexpected problems due to residual messages in the various queues.

STEP SWUFOPD2: Deactivate subscriptions

With this scenario, no unpropagated changes are replicated to either the primary server or the secondary server. The data on the primary server must be reinitialized with the data on the secondary server. This is achieved by first deactivating all the subscriptions, and then activating all the subscriptions using an automatic load (loadphase ‘I’) or a NO load (loadphase ‘N’).

- If the automatic load option is chosen, follow “STEP SWUFOPD2Ia: Deactivate subscriptions loadphase 'I'” on page 98 and “STEP SWUFOPB10Ib: Deactivate subscriptions loadphase 'I'” on page 79 before executing “STEP SWUFOPD3: Cold start Q Capture on both servers” on page 98.
- If the NO load option is chosen, follow “STEP SWUFOPD2Na: Deactivate subscriptions loadphase 'N'” on page 98, “STEP SWUFOPD2Nb: Stop workload on secondary” on page 98, and “STEP SWUFOPD2Nc: Resync database on primary server from secondary” on page 98, before executing “STEP SWUFOPD3: Cold start Q Capture on both servers” on page 98.

STEP SWUFOPD2Ia: Deactivate subscriptions loadphase 'I'

For the automatic load option, subscriptions can be deactivated as described in “STEP SWUFOPC10Ia: Deactivate subscriptions loadphase ‘I’” on page 88.

STEP SWUFOPD2Na: Deactivate subscriptions loadphase 'N'

For the NO load option, subscriptions can be deactivated as described in “STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'” on page 81.

STEP SWUFOPD2Nb: Stop workload on secondary

In this step, stop the application workload directed to this server without redirecting it to the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP SWUFOPD2Nc: Resync database on primary server from secondary

Since the NO load option is chosen, it is your responsibility to unload data from the secondary server and restore it on the primary server. We do not recommend any particular approach to achieve this function.

STEP SWUFOPD3: Cold start Q Capture on both servers

Cold start Q Capture on the primary and secondary server, as described in “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 82.

STEP SWUFOPD4: Start Q Apply on both servers

Start Q Apply on the primary server as described in “STEP SWCF1: Start Q Apply on primary” on page 47, and on the secondary server as described in “STEP SWCF4: Start Q Apply on secondary” on page 51.

STEP SWUFOPD5: Verify Q replication up and running on both servers

Verify that Q replication is up and running on both servers, as described in “STEP SWUFOPA17: Verify that Q replication up and running” on page 70.

STEP SWUFOPD6: Loadphase ‘I’ or ‘N’

The next steps to be executed depend upon whether the loadphase is ‘I’ or ‘N’, as follows:

- For loadphase ‘I’ follow “STEP SWUFOPD6Ia: Verify Q Capture latency and E2E latency” on page 99 through “STEP SWUFOPD6Ig: Start Q Capture on secondary” on page 99.
- For loadphase ‘N’ execute “STEP SWUFOPD6Na: Resume primary workload” on page 99.

STEP SWUFOPD6Ia: Verify Q Capture latency and E2E latency

In order to minimize the window during which the business application is not available to end users, it is desirable to shut it down on the secondary server after most of the unpropagated changes have been replicated over to the primary server, as described in “STEP SWCF5: Verify Q Capture latency and E2E latency” on page 51.

STEP SWUFOPD6Ib: Stop workload on secondary

In this step, stop the application workload directed to this server without redirecting it to the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the start of the outage of the business application.

STEP SWUFOPD6Ic: Autostop Q Capture on secondary

In this step, the Q Capture program on the secondary server is directed to shut down after it has completed processing all the changes that have been captured while the workload was running, and written it to the send queue, as described in “STEP SWCF8: Autostop Q Capture on secondary” on page 56.

STEP SWUFOPD6Ic: Verify XMITQ on secondary consumed

Verify that the transmit queue on the secondary server is fully consumed, as described in “STEP SWCF9: Verify XMITQ on secondary consumed” on page 57.

STEP SWUFOPD6Id: Verify RECVQ on primary consumed

Verify that the receive queue on the primary server is fully consumed, as described in “STEP SWCF10: Verify RECVQ on primary consumed” on page 57.

STEP SWUFOPD6Ie: Resume primary workload

Start the application workload on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

STEP SWUFOPD6If: Resume RO workload on secondary

The read-only workload can now be resumed on the secondary server.

STEP SWUFOPD6Ig: Start Q Capture on secondary

Start Q Capture on the secondary server, as described in “STEP SWCF3: Start Q Capture on secondary” on page 50.

STEP SWUFOPD6Na: Resume primary workload

When the loadphase is ‘N’, start the application workload on the primary server. The process used to achieve this varies from organization to organization and is beyond the scope of this book.

This is the end of the outage of the business application.

STEP SWUFOPD6Nb: Resume RO workload on secondary

The read-only workload can now be resumed on the secondary server.

Remove all remnants of Q replication objects on a server

In a disaster recovery scenario, the primary server site is permanently lost and the secondary server becomes the new primary server for the organization's business processes. Re-establishing the Q replication environment requires the following steps to be executed.

- Remove all remnants of Q replication objects, including WebSphere MQ channels, transmit queues, send queues, receive queues, restart queues, and admin queues, by submitting the JCL shown in Example 3-53. Step CLEARMQ does this.

  Note: The queue manager is not deleted since you will probably use the same one in the future.

- Remove all remnants of the Q replication Capture and Apply control tables by submitting the JCL shown in Example 3-53. Step DROPQCTL does this.

  Note: If you have installed the Q Monitor tables on this server, then you should drop them as well.

- Remove Q Capture and Q Apply log files. It is advisable to delete the Q Capture and Q Apply log files as a precautionary measure, as well submit the JCL shown in Example 3-53. Step DELQLOGS does this.

Example 3-53 JCL to remove all remnants of Q replication objects on the secondary server SC59

```
//TGTCLEAN JOB (POK,999),HUEYKIM,MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M
//*
//*************************************************************/
//*    Job to clean out all remnants of Q-Replication           */
//*    from the target system                                   */
//*                                                             */
//*    Step CLEARMQ: removes all MQ objects used for Q-Rep      */
//*        *** if there are any SPILLQs, add them to the list   */
//*                                                             */
//*    Step DELQLOGS: removes Q-APPLY and Q_CAPTURE logs.       */
//*                                                             */
//*    Step DROPQCTL: drop Q-replication control tables.        */
//*                                                             */
//*************************************************************/
//***---------------------------------------------------------***/
//CLEARMQ  EXEC PGM=CSQUTIL,PARM='MQ59'
//STEPLIB   DD DSN=MQ59.USERAUTH,DISP=SHR
//          DD DSN=MQ600.SCSQANLE,DISP=SHR
//          DD DSN=MQ600.SCSQAUTH,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *
STOP   CHANNEL(MQZ1.TO.MQ59)
STOP   CHANNEL(MQ59.TO.MQZ1)
DELETE CHANNEL(MQ59.TO.MQZ1)
DELETE CHANNEL(MQZ1.TO.MQ59)
```

```
CLEAR  QLOCAL(MQZ1XMIT)
CLEAR  QLOCAL(QREP.POKB.ADMINQ)
CLEAR  QLOCAL(QREP.POKB.RESTARTQ)
CLEAR  QLOCAL(QREP.POKA.TO.POKB.RECVQ)
DELETE QLOCAL(MQZ1XMIT)
DELETE QLOCAL(QREP.POKB.ADMINQ)
DELETE QLOCAL(QREP.POKB.RESTARTQ)
DELETE QLOCAL(QREP.POKA.TO.POKB.RECVQ)
DELETE QREMOTE(QREP.POKA.ADMINQ)
DELETE QREMOTE(QREP.POKB.TO.POKA.SENDQ)
DELETE QMODEL('IBMQREP.SPILL.MODELQ')
/*
//***----------------------------------------------------------***/
//*
//DELQLOGS EXEC PGM=IEFBR14
//DD1       DD DSN=HUEYKIM.DB8A.ITSO.QAPP.LOG,DISP=(MOD,DELETE),
//          UNIT=SYSDA,SPACE=(TRK,1)
//DD2       DD DSN=HUEYKIM.DB8A.ITSO.QCAP.LOG,DISP=(MOD,DELETE),
//          UNIT=SYSDA,SPACE=(TRK,1)
//*
//***----------------------------------------------------------***/
//DROPQCTL EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD DSN=DB8A8.SDSNEXIT,DISP=SHR
//         DD DSN=DB8A8.SDSNLOAD,DISP=SHR
//*         DD DSN=DB8GU.RUNLIB.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(DB8A)
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP81)  -
    LIB('DB8AU.RUNLIB.LOAD')
END
/*
//SYSIN  DD *
--
--DROP Q-REPLICATION CONTROL TABLES
--
DROP TABLE ITSO.IBMQREP_ADMINMSG;
DROP TABLE ITSO.IBMQREP_APPLYENQ;
DROP TABLE ITSO.IBMQREP_APPLYMON;
DROP TABLE ITSO.IBMQREP_APPLYPARMS;
DROP TABLE ITSO.IBMQREP_APPLYTRACE;
DROP TABLE ITSO.IBMQREP_CAPENQ;
DROP TABLE ITSO.IBMQREP_CAPMON;
DROP TABLE ITSO.IBMQREP_CAPPARMS;
DROP TABLE ITSO.IBMQREP_CAPQMON;
DROP TABLE ITSO.IBMQREP_CAPTRACE;
DROP TABLE ITSO.IBMQREP_DONEMSG;
DROP TABLE ITSO.IBMQREP_EXCEPTIONS;
DROP TABLE ITSO.IBMQREP_RECVQUEUES;
DROP TABLE ITSO.IBMQREP_SAVERI;
DROP TABLE ITSO.IBMQREP_SENDQUEUES;
DROP TABLE ITSO.IBMQREP_SIGNAL;
DROP TABLE ITSO.IBMQREP_SPILLEDROW;
DROP TABLE ITSO.IBMQREP_SPILLQS;
DROP TABLE ITSO.IBMQREP_SRCH_COND;
DROP TABLE ITSO.IBMQREP_SRC_COLS;
DROP TABLE ITSO.IBMQREP_SUBS;
DROP TABLE ITSO.IBMQREP_TARGETS;
DROP TABLE ITSO.IBMQREP_TRG_COLS;
--
--CHECK THAT ALL CONTROL TABLES HAVE BEEN DROPPED.
--
SELECT *
  FROM SYSIBM.SYSTABLES
 WHERE NAME LIKE 'IBMQREP%' AND CREATOR = 'ITSO';
/*
//***---------------------------------------------------------***/
```

Attention: After all remnants of Q replication objects have been removed from this server, re-configure this server for a new bidirectional Q replication environment where this server becomes the primary server and a new server is designated as the secondary server. For details on establishing a new bidirectional Q replication environment, refer to the IBM Redbook *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487.

4

WebSphere MQ shared queues and unidirectional replication

In this chapter we describe a step-by-step approach to implementing a WebSphere MQ shared queues high availability environment on the source system of a unidirectional Q replication environment.

The topics covered include:

- Business requirement
- Rationale for the unidirectional solution
- Environment configuration
- Set up of the environment
- Test cases

4.1 Introduction

Unidirectional replication using Q replication may be the replication of choice for environments that require a high-volume, low-latency solution between one or more tables on two servers, with the ability to perform transformations on the target tables. Such scenarios are typical of:

- Data warehousing applications that require simple to complex data transformations of operational data.
- Offloading of query reporting away from critical operational systems, by creating complete or partial subsets of operational data with or without transformations.

In this chapter we describe an existing WebSphere MQ shared queues high availability business solution implementation, into which is seamlessly introduced a unidirectional Q replication application. This unidirectional Q replication application coexists synergistically with the WebSphere MQ shared queues high availability solution. A brief overview of the business requirement is followed by a description of the environment configuration. The setup of this WebSphere MQ shared queues HA environment with unidirectional replication is described, followed by test cases showing the successful operation of the application in this environment. The topics covered in the following sections are:

- Business requirement
- Rationale for the unidirectional solution
- Environment configuration
- Set up of this environment
- Test cases

4.2 Business requirement

Our fictitious company StandOut is a fashion clothes retail organization with stores spread across the entire continental United States. It also provides its patrons with round-the-clock online shopping facilities. The requirement to provide high availability was met with a DB2 for z/OS data sharing implementation with WebSphere MQ shared queues for their catalog search and order processing application. The increase in the query workload by buyers in the various stores against the existing sales reporting application prompted a need to offload as much of the workload as possible off the critical operational systems.

These requirements may be summarized as follows:

1. Offload query workload against the existing sales reporting application to secondary servers.
2. Devise a solution that coexists synergistically with the existing operational systems' WebSphere MQ shared queues implementation involving DB2 for z/OS data sharing.

4.3 Rationale for the unidirectional solution

IBM's WebSphere Replication Server for z/OS provides the functionality of high-volume low-latency unidirectional replication for DB2 for z/OS data sources in a data sharing environment. The WebSphere Replication Server for z/OS use of WebSphere MQ as the underlying transport mechanism is particularly synergistic with the existing HA environment that uses WebSphere MQ shared queues.

4.4 Environment configuration

Figure 4-1 shows the configuration used in the StandOut unidirectional replication topology.

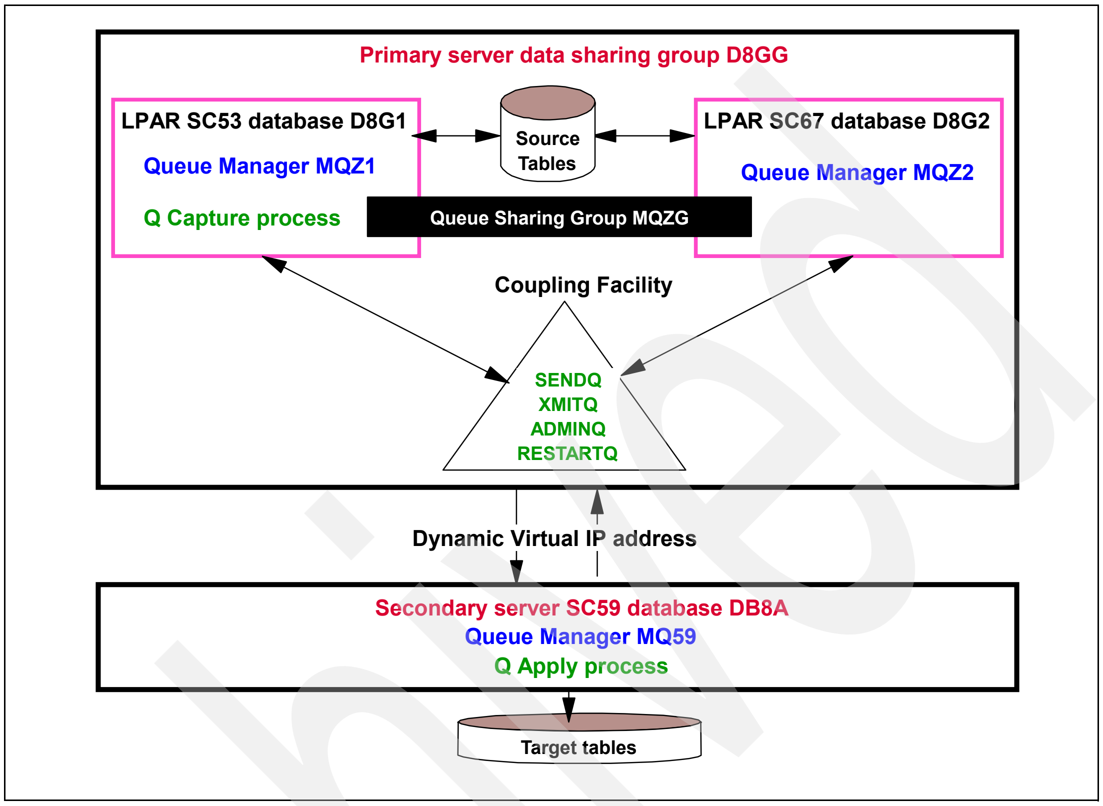


Figure 4-1 Unidirectional replication topology configuration

The primary server is logically a data sharing group (D8GG) involving two LPARs, SC53 (host name WTSC53.pok.ibm.com and IP address 9.12.6.77) and SC67, that are part of a two-member data sharing group. The source database is D8G1 on LPAR SC53 and D8G2 on LPAR SC67. The two different queue managers on each member of the queue-sharing group MQZG are MQZ1 and MQZ2. The Q Capture process is started on LPAR SC53. The Q replication queue objects such as the send queue (SENDQ), transmit queue (XMITQ), restart queue (RESTARTQ), and admin queue (ADMINQ) are defined in the coupling facility.

The secondary server SC59 has the host name WTSC59.pok.ibm.com and IP address 9.12.4.10. The target database is DB8A, and the queue manager is MQ59. The Q Apply process is started on this server. The Q replication queue objects such as the receive queue (RECVQ) are defined on this server.

Attention: A dynamic virtual IP address (DVIPA) is used in the definition of the channels between the secondary server and the primary server. By providing IP addresses for the z/OS applications that are not associated with a specific physical network attachment or gateway, a Virtual IP Address (VIPA) enables fault tolerance against outages in the IP interfaces on the z/OS host. Dynamically activated VIPAs are called DVIPAs. Using a DVIPA eliminates the need to modify the channel definition's IP address after LPAR SC53 fails, and Q Capture is started on LPAR 67 in order to continue normal replication operations. For more details on dynamic virtual IP address, refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance,* SG24-6517, and "IBM z/OS basic skills information center - Networking on z/OS":

http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.znetwork.doc/znetwork_5.html

http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.znetwork.doc/znetwork.pdf

We installed a set of two tables on the primary server, and replicated them to three tables on the secondary server, as shown in Figure 4-2. The sales table is split into two tables on the secondary server—one that contains sales for the eastern region, while the other contains sales for the western region. The DDL used in creating these tables is shown in Example 4-1 and Example 4-2.

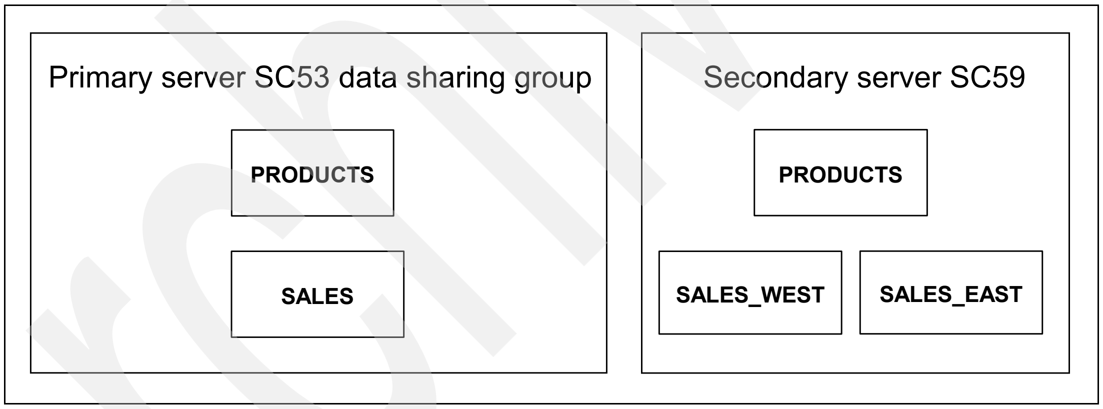


Figure 4-2 Tables used in the unidirectional replication scenario

Example 4-1 DDL of tables used on primary server SC53 in the unidirectional replication scenario

```
CREATE TABLE ITSOQ.PRODUCTS  (
            PRODUCT_ID       INTEGER NOT NULL
          , PRODUCT_NAME     VARCHAR(20) NOT NULL
          , PRODUCT_CATEGORY VARCHAR(10) NOT NULL
          , PRICE            DECIMAL(15, 2) NOT NULL WITH DEFAULT
          , MANUFACT_ID      INTEGER NOT NULL
          , PRIMARY KEY (PRODUCT_ID)
       )
       DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSOQ.PRODUCT_PK ON ITSOQ.PRODUCTS (PRODUCT_ID);


CREATE TABLE ITSOQ.SALES  (
            PRODUCT_ID       INTEGER NOT NULL
          , STORE_ID         INTEGER NOT NULL
          , QUANTITY_SOLD    INTEGER NOT NULL
          , DATE_OF_SALE     DATE NOT NULL
          , REGION_CODE      CHAR(10) NOT NULL
          , PRIMARY KEY (PRODUCT_ID, STORE_ID, DATE_OF_SALE)
        )
        DATA CAPTURE CHANGES;


CREATE TYPE 2 UNIQUE INDEX ITSOQ.SALES_PK ON ITSOQ.SALES (PRODUCT_ID, STORE_ID,
DATE_OF_SALE);
```

Example 4-2 DDL of the tables used on the secondary server SC59 in the unidirectional replication scenario

```
CREATE TABLE ITSOQ.PRODUCTS  (
            PRODUCT_ID       INTEGER NOT NULL
          , PRODUCT_NAME     VARCHAR(20) NOT NULL
          , PRODUCT_CATEGORY VARCHAR(10) NOT NULL
          , PRICE            DECIMAL(15, 2) NOT NULL WITH DEFAULT
          , MANUFACT_ID      INTEGER NOT NULL
          , PRIMARY KEY (PRODUCT_ID)
        )
        DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSOQ.PRODUCT_PK ON ITSOQ.PRODUCTS (PRODUCT_ID);


CREATE TABLE ITSOQ.SALES_EAST  (
            PRODUCT_ID       INTEGER NOT NULL
          , STORE_ID         INTEGER NOT NULL
          , QUANTITY_SOLD    INTEGER NOT NULL
          , DATE_OF_SALE     DATE NOT NULL
          , PRIMARY KEY (PRODUCT_ID, STORE_ID, DATE_OF_SALE)
        )
        DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSOQ.SALES_EAST_PK ON ITSOQ.SALES_EAST (PRODUCT_ID, STORE_ID,
DATE_OF_SALE);


CREATE TABLE ITSOQ.SALES_WEST  (
            PRODUCT_ID       INTEGER NOT NULL
          , STORE_ID         INTEGER NOT NULL
          , QUANTITY_SOLD    INTEGER NOT NULL
          , DATE_OF_SALE     DATE NOT NULL
          , PRIMARY KEY (PRODUCT_ID, STORE_ID, DATE_OF_SALE)
        )
        DATA CAPTURE CHANGES;

CREATE TYPE 2 UNIQUE INDEX ITSOQ.SALES_WEST_PK ON ITSOQ.SALES_WEST (PRODUCT_ID, STORE_ID,
DATE_OF_SALE);
```

Figure 4-3 provides a high-level overview of the various objects involved in implementing a unidirectional replication topology for StandOut. In Figure 4-3, there appear to be two sets of transmission queues, and sender and receiver channels on each server. However, there is only set on each server, as can be deduced from the identical names. Figure 4-3 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

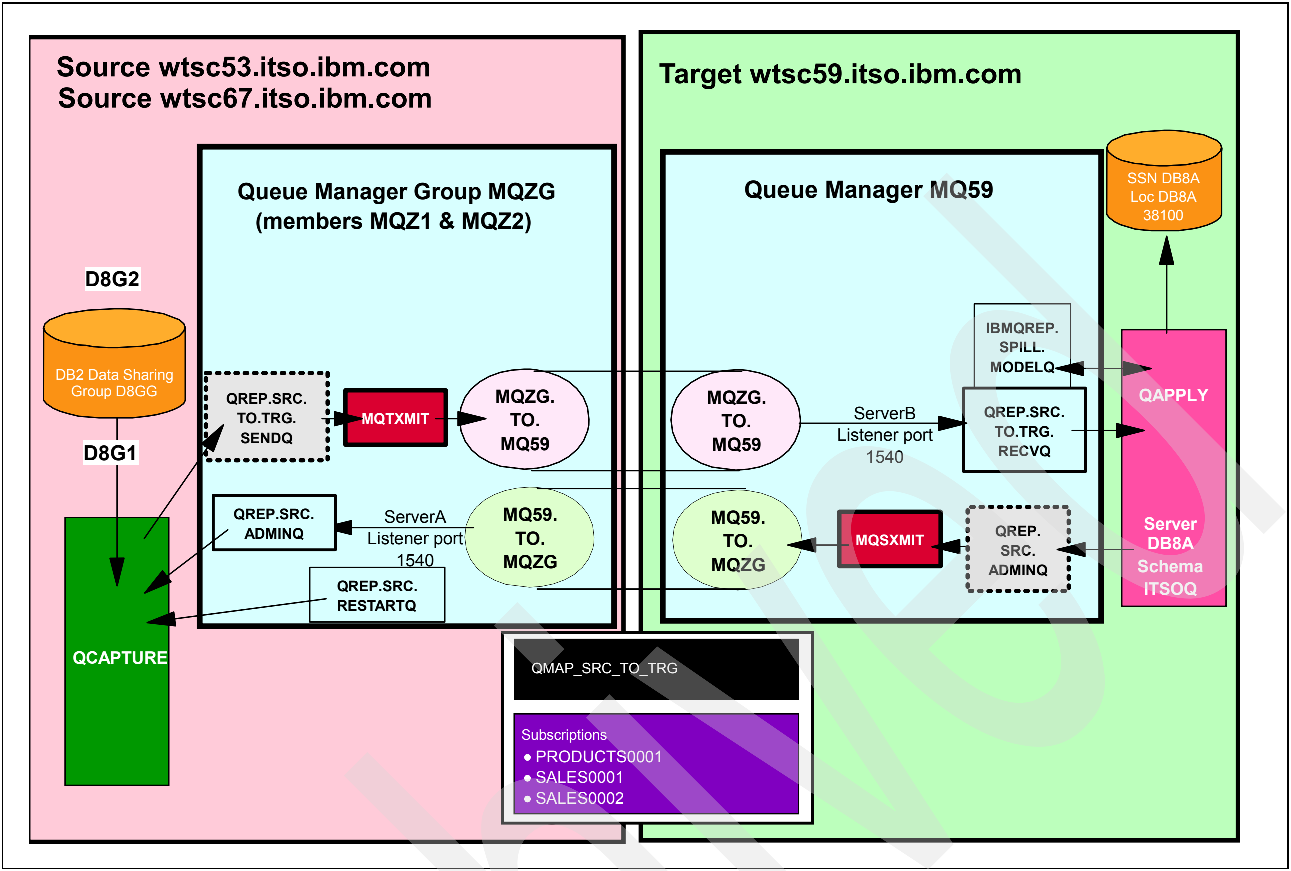


Figure 4-3 Unidirectional replication topology objects overview

4.5 Setup of this environment

Figure 4-4 provides an overview of the steps involved in setting up the unidirectional Q replication environment described in Figure 4-1, Figure 4-2, and Figure 4-3. The scripts used can be downloaded from the IBM Redbooks Web site:

`ftp://www.redbooks.ibm.com/redbooks/SG247215/`

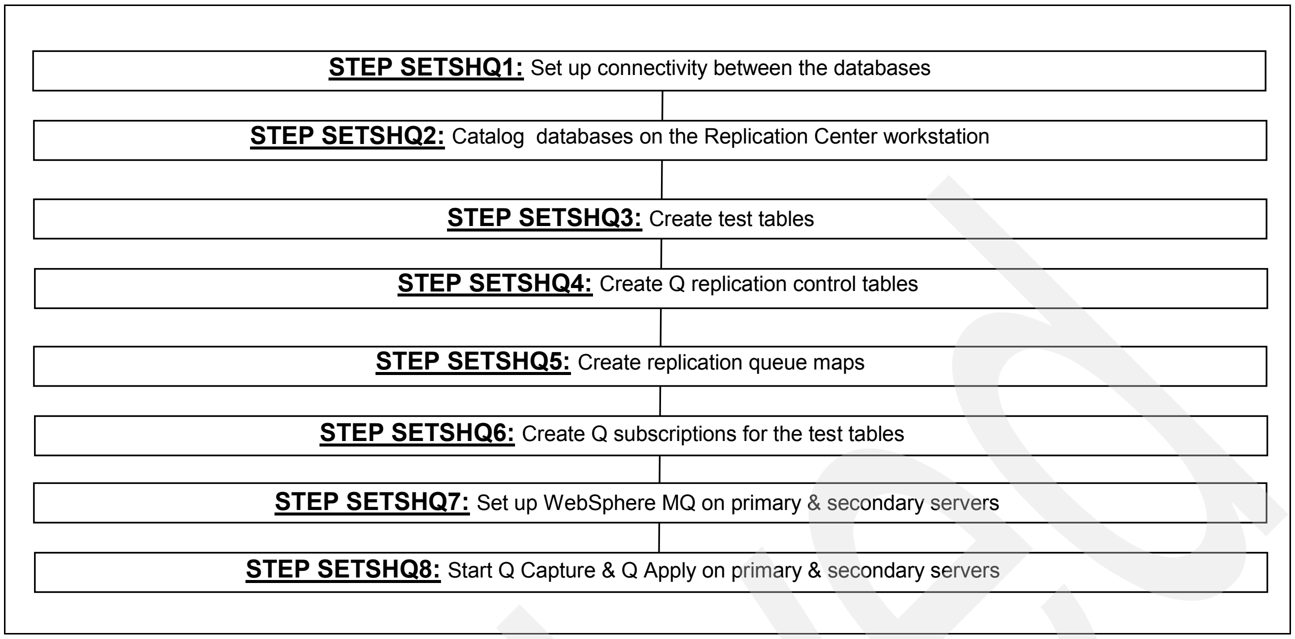

STEP SETSHQ1: Set up connectivity between the databases
STEP SETSHQ2: Catalog databases on the Replication Center workstation
STEP SETSHQ3: Create test tables
STEP SETSHQ4: Create Q replication control tables
STEP SETSHQ5: Create replication queue maps
STEP SETSHQ6: Create Q subscriptions for the test tables
STEP SETSHQ7: Set up WebSphere MQ on primary & secondary servers
STEP SETSHQ8: Start Q Capture & Q Apply on primary & secondary servers

Figure 4-4 Set up of the WebSphere shared queues unidirectional environment

Important: The steps described in Figure 4-4 assume that the shared queue environment has already been set up. Appendix C.1, “WebSphere shared queues overview” on page 252, provides an overview of shared queues.

Attention: For our scenario shown in Figure 4-1, the CF structures used by the queuing shared group MQZG are shown in Example 4-3. The CFSTRUCT parameter is supported for local and model queues and specifies the name of the coupling facility structure where you want messages stored when you use shared queues.

Example 4-4 shows the job for managing the queue-sharing group and queue manager definitions using the CSQ5PQSG utility. The data sharing group is DB8GU, while the local DB2 name is D8GG.

Example 4-5 shows the JCL for link-editing the various modules.

Example 4-3 Coupling facility structures used in defining the shared queue group MQZG

```
/*-------------------------------------------------------*/
/* MQZG STRUCTURES for coupling facility                 */
/*-------------------------------------------------------*/
  STRUCTURE NAME(MQZGCSQ_ADMIN)
        INITSIZE(10240)
        SIZE(20480)
        PREFLIST(CF03,CF06)
        REBUILDPERCENT(5)
        FULLTHRESHOLD(85)

  STRUCTURE NAME(MQZGAPPLICATION1)
         INITSIZE(10240)
         SIZE(20480)
         PREFLIST(CF03,CF06)
         REBUILDPERCENT(5)
         FULLTHRESHOLD(85)
```

Example 4-4 The JCL to define the MQZG queue-sharing group and queue managers MQZ1, MQZ2 to DB2

```
//MQZGAQS JOB (999,POK),'AQS',CLASS=A,MSGCLASS=T,
// REGION=0M,TIME=1440,NOTIFY=&SYSUID
/*JOBPARM L=9999,SYSAFF=SC53
//*               IBM WebSphere MQ for z/OS                         *
//*                                                                 *
//* Sample job to add a queue-sharing group record into the         *
//* DB2 administration table CSQ.ADMIN_B_QSG used by WebSphere MQ   *
//* using the CSQ5PQSG utility.                                     *
//*                                                                 *
//*******************************************************************
//*                                                                 *
//* MORE INFORMATION - See:                                         *
//*   "WebSphere MQ for z/OS System Setup Guide"                    *
//*     for information about this customization job                *
//*   "WebSphere MQ for z/OS System Administration Guide"           *
//*     for information about CSQ5PQSG                              *
//*     and managing queue-sharing groups                           *
//*                                                                 *
//*******************************************************************
//*******************************************************************
//*
//ADDQSG   EXEC PGM=CSQ5PQSG,REGION=4M,
//         PARM='ADD QSG,MQZG,DB8GU,D8GG'
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DSN=MQ531.SCSQANLE,DISP=SHR
//         DD DSN=MQ531.SCSQAUTH,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//
//ADDQMGR1 EXEC PGM=CSQ5PQSG,REGION=4M,
// PARM='ADD QMGR,MQZ1,MQZG,DB8GU,D8GG'
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DSN=MQ531.SCSQANLE,DISP=SHR
//         DD DSN=MQ531.SCSQAUTH,DISP=SHR
//         DD DSN=DB8F8.SDSNLOAD,DISP=SHR
//ADDQMGR2 EXEC PGM=CSQ5PQSG,REGION=4M,
// PARM='ADD QMGR,MQZ2,MQZG,DB8GU,D8GG'
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DSN=MQ531.SCSQANLE,DISP=SHR
//         DD DSN=MQ531.SCSQAUTH,DISP=SHR
//         DD DSN=DB8F8.SDSNLOAD,DISP=SHR
```

Example 4-5 JCL to link edit the CSQ6LOGP, CSQ6ARVP, and CSQ6SYSP modules

```
//MQZGZ1 JOB (999,POK),'MQ SERIES V5R3M1',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M                                   00020000
/*JOBPARM L=999,SYSAFF=SC42                                             00030001
//*CSQ4ZPRM JOB
//*******************************************************************
//*                                                                 *
//* @START_COPYRIGHT@                                             *
//* Statement:     Licensed Materials - Property of IBM           *
//*                                                               *
//*                5655-F10                                       *
//*                (C) Copyright IBM Corporation. 1993, 2002     *
//*                                                              *
//* Status:        Version 5 Release 3                           *
//* @END_COPYRIGHT@                                              *
//*                                                                *
//*******************************************************************
//*               IBM WebSphere MQ for z/OS                        *
//*
//*    This job assembles and links a new system parameter module.
//*
//*    Edit the parameters for the
//*    CSQ6LOGP, CSQ6ARVP, and CSQ6SYSP macros
//*    to determine your system parameters.
//*
//*
//*             Assemble step for CSQ6LOGP
//*
//LOGP   EXEC PGM=ASMA90,PARM='DECK,NOOBJECT,LIST,XREF(SHORT)',
//             REGION=4M
//SYSLIB   DD DSN=MQ531.SCSQMACS,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD UNIT=VIO,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&LOGP,
//            UNIT=VIO,DISP=(,PASS),
//            SPACE=(400,(100,100,1))
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
         CSQ6LOGP INBUFF=60,         ARCHIVE LOG BUFFER SIZES (KB)     X
               OUTBUFF=4000,          - INPUT AND OUTPUT               X
               MAXRTU=2,             MAX ALLOCATED ARCHIVE LOG UNITS   X
               DEALLCT=0,            ARCHIVE LOG DEALLOCATE INTERVAL   X
               OFFLOAD=NO,           ARCHIVING ACTIVE                  X
               MAXARCH=500,          MAX ARCHIVE LOG VOLUMES           X
               TWOACTV=NO,           DUAL ACTIVE LOGGING               X
               TWOARCH=NO,           DUAL ARCHIVE LOGGING              X
               TWOBSDS=YES,          DUAL BSDS                         X
               WRTHRSH=15            ACTIVE LOG BUFFERS
         END
/*
//*
//*             Assemble step for CSQ6ARVP
//*
//ARVP   EXEC PGM=ASMA90,COND=(0,NE),
//             PARM='DECK,NOOBJECT,LIST,XREF(SHORT)',
//             REGION=4M
//SYSLIB   DD DSN=MQ531.SCSQMACS,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD UNIT=VIO,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&ARVP,
//            UNIT=VIO,DISP=(,PASS),
//            SPACE=(400,(100,100,1))
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
         CSQ6ARVP ALCUNIT=BLK,       UNITS FOR PRIQTY/SECQTY           X
               ARCPFX1=CSQARC1,      DSN PREFIX FOR ARCHIVE LOG 1      X
               ARCPFX2=CSQARC2,      DSN PREFIX FOR ARCHIVE LOG 2      X
               ARCRETN=9999,         ARCHIVE LOG RETENION (DAYS)       X
               ARCWRTC=(1,3,4),      ARCHIVE WTO ROUTE CODE            X
               ARCWTOR=YES,          PROMPT BEFORE ARCHIVE LOG MOUNT   X
               BLKSIZE=28672,        ARCHIVE LOG BLOCKSIZE             X
```

```
                 CATALOG=NO,           CATALOG ARCHIVE LOG DATA SETS     X
                 COMPACT=NO,           ARCHIVE LOGS COMPACTED            X
                 PRIQTY=4320,          PRIMARY SPACE ALLOCATION          X
                 PROTECT=NO,           DISCRETE SECURITY PROFILES        X
                 QUIESCE=5,            MAX QUIESCE TIME (SECS)           X
                 SECQTY=540,           SECONDARY SPACE ALLOCATION        X
                 TSTAMP=NO,            TIMESTAMP SUFFIX IN DSN           X
                 UNIT=TAPE,            ARCHIVE LOG DEVICE TYPE 1         X
                 UNIT2=                ARCHIVE LOG DEVICE TYPE 2
           END
/*
//*
//*              Assemble step for CSQ6SYSP
//*
//SYSP   EXEC PGM=ASMA90,COND=(0,NE),
//             PARM='DECK,NOOBJECT,LIST,XREF(SHORT)',
//             REGION=4M
//SYSLIB   DD DSN=MQ531.SCSQMACS,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD UNIT=VIO,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&SYSP,
//            UNIT=VIO,DISP=(,PASS),
//            SPACE=(400,(100,100,1))
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
         CSQ6SYSP CTHREAD=1300,      TOTAL NUMBER OF CONNECTIONS       X
               CMDUSER=CSQOPR,       DEFAULT USERID FOR COMMANDS       X
               EXITLIM=30,           EXIT TIMEOUT (SEC)                X
               EXITTCB=8,            NUMBER OF EXIT SERVER TCBS        X
               IDBACK=600,           NUMBER OF NON-TSO CONNECTIONS     X
               IDFORE=600,           NUMBER OF TSO CONNECTIONS         X
               LOGLOAD=500000,       LOG RECORD CHECKPOINT NUMBER      X
               OTMACON=(,,DFSYDRU0,2147483647,CSQ),  OTMA PARAMETERS   X
               QINDXBLD=WAIT,        QUEUE INDEX BUILDING              X
               QMCCSID=0,            QMGR CCSID                        X
               QSGDATA=(MQZG,DB8GU,D8GG,10),                           X
               RESAUDIT=YES,         RESLEVEL AUDITING                 X
               ROUTCDE=1,            DEFAULT WTO ROUTE CODE            X
               SMFACCT=NO,           GATHER SMF ACCOUNTING             X
               SMFSTAT=NO,           GATHER SMF STATS                  X
               STATIME=30,           STATISTICS RECORD INTERVAL (MIN)  X
               TRACSTR=YES,          TRACING AUTO START                X
               TRACTBL=99,           GLOBAL TRACE TABLE SIZE X4K       X
               WLMTIME=30,           WLM QUEUE SCAN INTERVAL (SEC)     X
               SERVICE=0             IBM SERVICE USE ONLY
         END
/*
//*
//*  LINKEDIT ARVP, LOGP, and SYSP into a
//*  system parameter module.
//*
//LKED   EXEC PGM=IEWL,COND=(0,NE),
//      PARM='SIZE=(900K,124K),RENT,NCAL,LIST,AMODE=31,RMODE=ANY'
//*
//*   OUPUT AUTHORIZED APF LIBRARY FOR THE NEW SYSTEM
//*   PARAMETER MODULE.
//*
//SYSLMOD  DD DSN=MQZ1.USERAUTH,DISP=SHR
//SYSUT1   DD UNIT=VIO,DCB=BLKSIZE=1024,
//            SPACE=(1024,(200,20))
```

```
//SYSPRINT DD SYSOUT=*
//ARVP     DD DSN=&&ARVP,DISP=(OLD,DELETE)
//LOGP     DD DSN=&&LOGP,DISP=(OLD,DELETE)
//SYSP     DD DSN=&&SYSP,DISP=(OLD,DELETE)
//*
//*   LOAD LIBRARY containing the default system
//*   parameter module (CSQZPARM).
//*
//OLDLOAD  DD DSN=MQ531.SCSQAUTH,DISP=SHR
//SYSLIN   DD *
   INCLUDE SYSP
   INCLUDE ARVP
   INCLUDE LOGP
   INCLUDE OLDLOAD(CSQZPARM)
 ENTRY CSQZMSTR
 NAME CSQZPARM(R)                   Your system parameter module name
/*
//
```

Each of the steps described in Figure 4-4 is discussed briefly in the following sections.

4.5.1 STEP SETSHQ1: Set up connectivity between the databases

In this step connectivity is established between the source and target databases. This involves inserting rows into the SYSIBM.LOCATIONS, SYSIBM.IPNAMES, and SYSIBM.USERNAMES tables on the primary server SC53, as shown in Example 4-6, and secondary server SC59, as shown in Example 4-7.

Example 4-6 Create connectivity between the source and the target databases on the primary server SC53

```
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, IBMREQD, PORT, TPN)
VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
'DB8A  ' , -- LOCATION CHAR 16 NO
'DB8A ' , -- LINKNAME CHAR 8 NO
' ' , -- IBMREQD CHAR 1 NO
'38100 ' , -- PORT CHAR 32 NO
' ' ); -- TPN VARCHAR 64 NO

INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IBMREQD,IPADDR)
VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
'DB8A ' , -- LINKNAME CHAR 8 NO
'P' , -- SECURITY_OUT CHAR 1 NO
'O' , -- USERNAMES CHAR 1 NO
' ' , -- IBMREQD CHAR 1 NO
'9.12.4.10' ); -- IPADDR VARCHAR 254 NO

INSERT INTO SYSIBM.USERNAMES (TYPE, AUTHID, LINKNAME,
NEWAUTHID, PASSWORD,IBMREQD)
VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
'O' , -- TYPE CHAR 1 NO
' ' , -- AUTHID CHAR 8 NO
'DB8A  ' , -- LINKNAME CHAR 8 NO
'MWIMAN' , -- NEWAUTHID CHAR 8 NO
'MWIMAN' , -- PASSWORD CHAR 8 NO
' ' ); -- IBMREQD CHAR 1 NO
```

Example 4-7 Create connectiivity between target and source databases on the secondary server SC59

```
INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, IBMREQD, PORT, TPN)
VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
'DB8G  ' , -- LOCATION CHAR 16 NO
'D8GG ' , -- LINKNAME CHAR 8 NO
' ' , -- IBMREQD CHAR 1 NO
'38060 ' , -- PORT CHAR 32 NO
' ' ); -- TPN VARCHAR 64 NO

--
-- 9.12.6.11 is a DVIPA for both data sharing subsystems
-- if do not have DVIPA then would set ip address to one of
-- the subsystems such as D8G1 on 9.12.6.77 and then add the other
-- to the iplist such as D8G2 on 9.12.6.66
INSERT INTO SYSIBM.IPNAMES (LINKNAME, SECURITY_OUT, USERNAMES, IBMREQD,IPADDR)
VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
'D8GG ' , -- LINKNAME CHAR 8 NO
'P' , -- SECURITY_OUT CHAR 1 NO
'O' , -- USERNAMES CHAR 1 NO
' ' , -- IBMREQD CHAR 1 NO
'9.12.6.11' ); -- IPADDR VARCHAR 254 NO

--INSERT INTO SYSIBM.IPLIST (LINKNAME, IBMREQD,IPADDR)
--VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
--'D8GG ' , -- LINKNAME CHAR 8 NO
--' ' , -- IBMREQD CHAR 1 NO
--'9.12.6.66' ); -- IPADDR VARCHAR 254 NO

INSERT INTO SYSIBM.USERNAMES (TYPE, AUTHID, LINKNAME,
NEWAUTHID, PASSWORD,IBMREQD)
VALUES (
-- ENTER VALUES BELOW COLUMN NAME DATA TYPE LENGTH NULLS
'O' , -- TYPE CHAR 1 NO
' ' , -- AUTHID CHAR 8 NO
'D8GG  ' , -- LINKNAME CHAR 8 NO
'MWIMAN' , -- NEWAUTHID CHAR 8 NO
'MWIMAN' , -- PASSWORD CHAR 8 NO
' ' ); -- IBMREQD CHAR 1 NO
```

4.5.2 STEP SETSHQ2: Catalog databases in the Replication Center

To configure the Q replication environment from the Replication Center, you need to catalog the source and target databases, as shown in Example 4-8.

Example 4-8 Catalog the source and target databases in the Replication Center

```
db2 catalog tcpip node WTSC53 remote wtsc53.itso.ibm.com server 38060
db2 catalog db D8G1 at node WTSC53 authentication dcs
db2 catalog dcs db D8G1 as DB8G

db2 catalog tcpip node WTSC67 remote wtsc67.itso.ibm.com server 38060
db2 catalog db D8G2 at node WTSC67 authentication dcs
db2 catalog dcs db D8G2 as DB8G

db2 catalog db D8GG at node WTSC53 authentication dcs
db2 catalog dcs db D8GG as DB8G
```

```
db2 catalog tcpip node WTSC59 remote wtsc59.itso.ibm.com server 38100
db2 catalog db DB8A at node WTSC59 authentication dcs
db2 catalog dcs db DB8A as DB8A
```

4.5.3 STEP SETSHQ3: Create test tables

The test tables used in the scenario described in Figure 4-2 on page 106 are created using DDL, shown in Example 4-1 on page 106 and Example 4-2 on page 107.

4.5.4 STEP SETSHQ4: Create Q replication control tables

Example 4-9 and Example 4-10 show the DDL for creating the Q replication control tables on the primary server SC53 and secondary server SC59, respectively. This SQL was generated using the Replication Center GUI.

Example 4-9 Create Q replication control tables on the primary server SC53

```
--Beginning of script 1--   DatabaseDB2OS390 (D8G1) [WARNING***Please do not alter this
line]--
CONNECT TO D8G2 USER mwiman using mwiman;

CREATE TABLESPACE QCITSO2 IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CCSID EBCDIC;

CREATE TABLE ITSOQ.IBMQREP_CAPPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 REMOTE_SRC_SERVER VARCHAR(18),
 RESTARTQ VARCHAR(48) NOT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 STARTMODE VARCHAR(6) NOT NULL WITH DEFAULT 'WARMSI',
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 500,
 COMMIT_INTERVAL INTEGER NOT NULL WITH DEFAULT 500,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300000,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 SIGNAL_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 SLEEP_INTERVAL INTEGER NOT NULL WITH DEFAULT 5000,
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 CAPTURE_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0901',
 COMPATIBILITY CHARACTER(4) NOT NULL WITH DEFAULT '0901'
)
 IN QREPITSO.QCITSO2;

CREATE UNIQUE INDEX ITSOQ.IX1CQMGRCOL ON ITSOQ.IBMQREP_CAPPARMS
(
 QMGR ASC
);

CREATE TABLESPACE QCITSO3 IN QREPITSO
  SEGSIZE 4
  LOCKSIZE ROW
  CCSID EBCDIC;

CREATE TABLE ITSOQ.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME VARCHAR(128) NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 RECVQ VARCHAR(48),
 MESSAGE_FORMAT CHARACTER(1) NOT NULL WITH DEFAULT 'C',
 MSG_CONTENT_TYPE CHARACTER(1) NOT NULL WITH DEFAULT 'T',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'S',
 HEARTBEAT_INTERVAL INTEGER NOT NULL WITH DEFAULT 60,
 MAX_MESSAGE_SIZE INTEGER NOT NULL WITH DEFAULT 64,
 APPLY_SERVER VARCHAR(18),
 APPLY_ALIAS VARCHAR(8),
 APPLY_SCHEMA VARCHAR(128),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(SENDQ)
)
 IN QREPITSO.QCITSO3;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_SENDQUEUES ON
 ITSOQ.IBMQREP_SENDQUEUES
(
 SENDQ
);

CREATE UNIQUE INDEX ITSOQ.IX1PUBMAPCOL ON ITSOQ.IBMQREP_SENDQUEUES
(
 PUBQMAPNAME ASC
);

CREATE TABLE ITSOQ.IBMQREP_SUBS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 TARGET_SERVER VARCHAR(18),
 TARGET_ALIAS VARCHAR(8),
 TARGET_OWNER VARCHAR(128),
 TARGET_NAME VARCHAR(128),
 TARGET_TYPE INTEGER,
 APPLY_SCHEMA VARCHAR(128),
 SENDQ VARCHAR(48) NOT NULL,
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 ALL_CHANGED_ROWS CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 BEFORE_VALUES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 CHANGED_COLS_ONLY CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 STATE_TRANSITION VARCHAR(256) FOR BIT DATA,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_MEMBERS CHARACTER(254) FOR BIT DATA WITH DEFAULT NULL,
 OPTIONS_FLAG CHARACTER(4) NOT NULL WITH DEFAULT 'NNNN',
 SUPPRESS_DELETES CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 DESCRIPTION VARCHAR(200),
 TOPIC VARCHAR(256),
 PRIMARY KEY(SUBNAME),
 CONSTRAINT FKSENDQ FOREIGN KEY(SENDQ) REFERENCES
 ITSOQ.IBMQREP_SENDQUEUES(SENDQ)
)
 IN QREPITSO.QCITSO3;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_SUBS ON ITSOQ.IBMQREP_SUBS
(
 SUBNAME
);

CREATE TABLE ITSOQ.IBMQREP_SRC_COLS
(
 SUBNAME VARCHAR(132) NOT NULL,
 SRC_COLNAME VARCHAR(128) NOT NULL,
 IS_KEY SMALLINT NOT NULL WITH DEFAULT 0,
 BEFORE_VALUE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 PRIMARY KEY(SUBNAME, SRC_COLNAME),
 CONSTRAINT FKSUBS FOREIGN KEY(SUBNAME) REFERENCES ITSOQ.IBMQREP_SUBS
(SUBNAME)
)
 IN QREPITSO.QCITSO2;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_SRC_COLS ON ITSOQ.IBMQREP_SRC_COLS
(
 SUBNAME,
 SRC_COLNAME
);

CREATE TABLE ITSOQ.IBMQREP_SRCH_COND
(
 ASNQREQD INTEGER
)
 IN QREPITSO.QCITSO2;

CREATE TABLE ITSOQ.IBMQREP_SIGNAL
(
 SIGNAL_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 SIGNAL_TYPE VARCHAR(30) NOT NULL,
 SIGNAL_SUBTYPE VARCHAR(30),
 SIGNAL_INPUT_IN VARCHAR(500),
 SIGNAL_STATE CHARACTER(1) NOT NULL WITH DEFAULT 'P',
 SIGNAL_LSN CHARACTER(10) FOR BIT DATA,
 PRIMARY KEY(SIGNAL_TIME)
)
 IN QREPITSO.QCITSO3
 DATA CAPTURE CHANGES;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_SIGNAL ON ITSOQ.IBMQREP_SIGNAL
(
 SIGNAL_TIME
);

CREATE TABLE ITSOQ.IBMQREP_CAPTRACE
```

```
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QCITS03;

CREATE TABLE ITSOQ.IBMQREP_CAPMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 CURRENT_LOG_TIME TIMESTAMP NOT NULL,
 CAPTURE_IDLE INTEGER NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 ROWS_PROCESSED INTEGER NOT NULL,
 TRANS_SKIPPED INTEGER NOT NULL,
 TRANS_PROCESSED INTEGER NOT NULL,
 TRANS_SPILLED INTEGER NOT NULL,
 MAX_TRANS_SIZE INTEGER NOT NULL,
 QUEUES_IN_ERROR INTEGER NOT NULL,
 RESTART_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 CURRENT_SEQ CHARACTER(10) FOR BIT DATA NOT NULL,
 PRIMARY KEY(MONITOR_TIME)
)
 IN QREPITSO.QCITS02;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_CAPMON ON ITSOQ.IBMQREP_CAPMON
(
 MONITOR_TIME
);

CREATE TABLE ITSOQ.IBMQREP_CAPQMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 SENDQ VARCHAR(48) NOT NULL,
 ROWS_PUBLISHED INTEGER NOT NULL,
 TRANS_PUBLISHED INTEGER NOT NULL,
 CHG_ROWS_SKIPPED INTEGER NOT NULL,
 DELROWS_SUPPRESSED INTEGER NOT NULL,
 ROWS_SKIPPED INTEGER NOT NULL,
 PRIMARY KEY(MONITOR_TIME, SENDQ)
)
 IN QREPITSO.QCITS02;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_CAPQMON ON ITSOQ.IBMQREP_CAPQMON
(
 MONITOR_TIME,
 SENDQ
);

CREATE TABLE ITSOQ.IBMQREP_CAPENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QCITS03;

CREATE TABLE ITSOQ.IBMQREP_ADMINMSG
(
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 MSG_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 PRIMARY KEY(MQMSGID)
)
 IN QREPITSO.QCITSO2;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_ADMINMSG ON ITSOQ.IBMQREP_ADMINMSG
(
 MQMSGID
);

INSERT INTO ITSOQ.IBMQREP_CAPPARMS
 (qmgr, restartq, adminq, startmode, memory_limit, commit_interval,
 autostop, monitor_interval, monitor_limit, trace_limit, signal_limit,
 prune_interval, sleep_interval, logreuse, logstdout, term, arch_level)
 VALUES
 ('MQZG', 'QREP.SRC.RESTARTQ', 'QREP.SRC.ADMINQ', 'WARMSI', 32, 500
, 'N', 300000, 10080, 10080, 10080, 300, 5000, 'N', 'N', 'Y', '0901');

COMMIT;
terminate;
```

Example 4-10   Create Q replication control tables on the secondary server SC59

```
--Beginning of script 1--   DatabaseDB2OS390 (DB8A) [WARNING***Please do not alter this
line]--

CONNECT TO DB8A USER hueykim using hueykim;
CREATE TABLESPACE QCITSO IN QREPITSO
  SEGSIZE 4
  LOCKSIZE PAGE
  CCSID EBCDIC;



CREATE TABLE ITSOQ.IBMQREP_APPLYPARMS
(
 QMGR VARCHAR(48) NOT NULL,
 MONITOR_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 TRACE_LIMIT INTEGER NOT NULL WITH DEFAULT 10080,
 MONITOR_INTERVAL INTEGER NOT NULL WITH DEFAULT 300000,
 PRUNE_INTERVAL INTEGER NOT NULL WITH DEFAULT 300,
 AUTOSTOP CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGREUSE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOGSTDOUT CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 APPLY_PATH VARCHAR(1040) WITH DEFAULT NULL,
 ARCH_LEVEL CHARACTER(4) NOT NULL WITH DEFAULT '0901',
 TERM CHARACTER(1) NOT NULL WITH DEFAULT 'Y',
 PWDFILE VARCHAR(48) WITH DEFAULT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL WITH DEFAULT 3,
 SQL_CAP_SCHEMA VARCHAR(128) WITH DEFAULT NULL
)
 IN QREPITSO.QCITSO;

CREATE UNIQUE INDEX ITSOQ.IX1AQMGRCOL ON ITSOQ.IBMQREP_APPLYPARMS
(
 QMGR ASC
);

CREATE TABLE ITSOQ.IBMQREP_RECVQUEUES
(
 REPQMAPNAME VARCHAR(128) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SENDQ VARCHAR(48) WITH DEFAULT NULL,
 ADMINQ VARCHAR(48) NOT NULL,
 NUM_APPLY_AGENTS INTEGER NOT NULL WITH DEFAULT 16,
 MEMORY_LIMIT INTEGER NOT NULL WITH DEFAULT 32,
 CAPTURE_SERVER VARCHAR(18) NOT NULL,
 CAPTURE_ALIAS VARCHAR(8) NOT NULL,
 CAPTURE_SCHEMA VARCHAR(30) NOT NULL WITH DEFAULT 'ASN',
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'A',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 DESCRIPTION VARCHAR(254),
 PRIMARY KEY(RECVQ)
)
 IN QREPITSO.QCITSO1;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_RECVQUEUES ON
 ITSOQ.IBMQREP_RECVQUEUES
(
 RECVQ
);

CREATE UNIQUE INDEX ITSOQ.IX1REPMAPCOL ON ITSOQ.IBMQREP_RECVQUEUES
(
 REPQMAPNAME ASC
);

CREATE TABLE ITSOQ.IBMQREP_TARGETS
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 SUB_ID INTEGER WITH DEFAULT NULL,
 SOURCE_SERVER VARCHAR(18) NOT NULL,
 SOURCE_ALIAS VARCHAR(8) NOT NULL,
 SOURCE_OWNER VARCHAR(128) NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SRC_NICKNAME_OWNER VARCHAR(128),
 SRC_NICKNAME VARCHAR(128),
 TARGET_OWNER VARCHAR(128) NOT NULL,
 TARGET_NAME VARCHAR(128) NOT NULL,
 TARGET_TYPE INTEGER NOT NULL WITH DEFAULT 1,
 FEDERATED_TGT_SRVR VARCHAR(18) WITH DEFAULT NULL,
 STATE CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 STATE_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 STATE_INFO CHARACTER(8),
 SUBTYPE CHARACTER(1) NOT NULL WITH DEFAULT 'U',
 CONFLICT_RULE CHARACTER(1) NOT NULL WITH DEFAULT 'K',
 CONFLICT_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'I',
 ERROR_ACTION CHARACTER(1) NOT NULL WITH DEFAULT 'Q',
 SPILLQ VARCHAR(48) WITH DEFAULT NULL,
 OKSQLSTATES VARCHAR(128) WITH DEFAULT NULL,
 SUBGROUP VARCHAR(30) WITH DEFAULT NULL,
 SOURCE_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 TARGET_NODE SMALLINT NOT NULL WITH DEFAULT 0,
 GROUP_INIT_ROLE CHARACTER(1) WITH DEFAULT NULL,
 HAS_LOADPHASE CHARACTER(1) NOT NULL WITH DEFAULT 'N',
 LOAD_TYPE SMALLINT NOT NULL WITH DEFAULT 0,
 DESCRIPTION VARCHAR(254),
 SEARCH_CONDITION VARCHAR(2048) WITH DEFAULT NULL,
 MODELQ VARCHAR(36) NOT NULL WITH DEFAULT 'IBMQREP.SPILL.MODELQ',
 CCD_CONDENSED CHARACTER(1) WITH DEFAULT 'Y',
 CCD_COMPLETE CHARACTER(1) WITH DEFAULT 'Y',
 BEFORE_IMG_PREFIX VARCHAR(4) WITH DEFAULT NULL
)
 IN QREPITSO.QCITSO1;

CREATE UNIQUE INDEX ITSOQ.IX1TARGETS ON ITSOQ.IBMQREP_TARGETS
(
 SUBNAME ASC,
 RECVQ ASC
);

CREATE UNIQUE INDEX ITSOQ.IX2TARGETS ON ITSOQ.IBMQREP_TARGETS
(
 TARGET_OWNER ASC,
 TARGET_NAME ASC,
 RECVQ ASC,
 SOURCE_OWNER ASC,
 SOURCE_NAME ASC
);

CREATE INDEX ITSOQ.IX3TARGETS ON ITSOQ.IBMQREP_TARGETS
(
 RECVQ ASC,
 SUB_ID ASC
);

CREATE TABLE ITSOQ.IBMQREP_TRG_COLS
(
 RECVQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 SOURCE_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNAME VARCHAR(128) NOT NULL,
 TARGET_COLNO INTEGER WITH DEFAULT NULL,
 MSG_COL_CODEPAGE INTEGER WITH DEFAULT NULL,
 MSG_COL_NUMBER SMALLINT WITH DEFAULT NULL,
 MSG_COL_TYPE SMALLINT WITH DEFAULT NULL,
 MSG_COL_LENGTH INTEGER WITH DEFAULT NULL,
 IS_KEY CHARACTER(1) NOT NULL,
 MAPPING_TYPE CHARACTER(1) WITH DEFAULT NULL,
 BEF_TARG_COLNAME VARCHAR(128) WITH DEFAULT null
)
 IN QREPITSO.QCITSO;

CREATE UNIQUE INDEX ITSOQ.IX1TRGCOL ON ITSOQ.IBMQREP_TRG_COLS
(
 RECVQ ASC,
 SUBNAME ASC,
 TARGET_COLNAME ASC
);

CREATE TABLE ITSOQ.IBMQREP_SPILLQS
(
 SPILLQ VARCHAR(48) NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 PRIMARY KEY(SPILLQ)
)
 IN QREPITSO.QCITSO;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_SPILLQS ON ITSOQ.IBMQREP_SPILLQS
```

```
(
 SPILLQ
);

CREATE LOB TABLESPACE LXTIBMQQ IN QREPITSO
  LOG NO;

CREATE TABLE ITSOQ.IBMQREP_EXCEPTIONS
(
 EXCEPTION_TIME TIMESTAMP NOT NULL WITH DEFAULT,
 RECVQ VARCHAR(48) NOT NULL,
 SRC_COMMIT_LSN VARCHAR(48) FOR BIT DATA NOT NULL,
 SRC_TRANS_TIME TIMESTAMP NOT NULL,
 SUBNAME VARCHAR(132) NOT NULL,
 REASON CHARACTER(12) NOT NULL,
 SQLCODE INTEGER,
 SQLSTATE CHARACTER(5),
 SQLERRMC VARCHAR(70) FOR BIT DATA,
 OPERATION VARCHAR(18) NOT NULL,
 TEXT CLOB(32768) NOT NULL,
 IS_APPLIED CHARACTER(1) NOT NULL,
 CONFLICT_RULE CHARACTER(1),
 REPLROWID ROWID NOT NULL GENERATED BY DEFAULT
)
 IN QREPITSO.QCITSO;

CREATE AUXILIARY TABLE ITSOQ.XTIBMQREP_EXCEPTIONSO
 IN QREPITSO.LXTIBMQQ
 STORES ITSOQ.IBMQREP_EXCEPTIONS COLUMN TEXT;

CREATE INDEX ITSOQ.XIXTIBMQREP_EXCEPTIONSO ON
 ITSOQ.XTIBMQREP_EXCEPTIONSO;

CREATE UNIQUE INDEX ITSOQ.RIIBMQREP_EXCEPTIONS ON
 ITSOQ.IBMQREP_EXCEPTIONS
(
 REPLROWID
);

CREATE TABLE ITSOQ.IBMQREP_APPLYTRACE
(
 OPERATION CHARACTER(8) NOT NULL,
 TRACE_TIME TIMESTAMP NOT NULL,
 DESCRIPTION VARCHAR(1024) NOT NULL
)
 IN QREPITSO.QCITSO1;

CREATE INDEX ITSOQ.IX1TRCTMCOL ON ITSOQ.IBMQREP_APPLYTRACE
(
 TRACE_TIME ASC
);

CREATE TABLE ITSOQ.IBMQREP_APPLYMON
(
 MONITOR_TIME TIMESTAMP NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 QSTART_TIME TIMESTAMP NOT NULL,
 CURRENT_MEMORY INTEGER NOT NULL,
 QDEPTH INTEGER NOT NULL,
 END2END_LATENCY INTEGER NOT NULL,
 QLATENCY INTEGER NOT NULL,
 APPLY_LATENCY INTEGER NOT NULL,
 TRANS_APPLIED INTEGER NOT NULL,
 ROWS_APPLIED INTEGER NOT NULL,
 TRANS_SERIALIZED INTEGER NOT NULL,
 RI_DEPENDENCIES INTEGER NOT NULL,
 RI_RETRIES INTEGER NOT NULL,
 DEADLOCK_RETRIES INTEGER NOT NULL,
 ROWS_NOT_APPLIED INTEGER NOT NULL,
 MONSTER_TRANS INTEGER NOT NULL,
 MEM_FULL_TIME INTEGER NOT NULL,
 APPLY_SLEEP_TIME INTEGER NOT NULL,
 SPILLED_ROWS INTEGER NOT NULL,
 SPILLEDROWSAPPLIED INTEGER NOT NULL,
 OLDEST_TRANS TIMESTAMP NOT NULL,
 OKSQLSTATE_ERRORS INTEGER NOT NULL,
 HEARTBEAT_LATENCY INTEGER NOT NULL,
 KEY_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_DEPENDENCIES INTEGER NOT NULL,
 UNIQ_RETRIES INTEGER NOT NULL,
 OLDEST_INFLT_TRANS TIMESTAMP,
 PRIMARY KEY(MONITOR_TIME, RECVQ)
)
 IN QREPITSO.QCITSO;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_APPLYMON ON ITSOQ.IBMQREP_APPLYMON
(
 MONITOR_TIME,
 RECVQ
);

CREATE TABLE ITSOQ.IBMQREP_DONEMSG
(
 RECVQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(RECVQ, MQMSGID)
)
 IN QREPITSO.QCITSO;

CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_DONEMSG ON ITSOQ.IBMQREP_DONEMSG
(
 RECVQ,
 MQMSGID
);

CREATE TABLE ITSOQ.IBMQREP_SPILLEDROW
(
 SPILLQ VARCHAR(48) NOT NULL,
 MQMSGID CHARACTER(24) FOR BIT DATA NOT NULL,
 PRIMARY KEY(SPILLQ, MQMSGID)
)
 IN QREPITSO.QCITSO;
CREATE UNIQUE INDEX ITSOQ.PKIBMQREP_SPILLEDROW ON
 ITSOQ.IBMQREP_SPILLEDROW
(
 SPILLQ,
 MQMSGID
);

CREATE TABLE ITSOQ.IBMQREP_SAVERI
```

```
(
 SUBNAME VARCHAR(132) NOT NULL,
 RECVQ VARCHAR(48) NOT NULL,
 CONSTNAME VARCHAR(128) NOT NULL,
 TABSCHEMA VARCHAR(128) NOT NULL,
 TABNAME VARCHAR(128) NOT NULL,
 REFTABSCHEMA VARCHAR(128) NOT NULL,
 REFTABNAME VARCHAR(128) NOT NULL,
 ALTER_RI_DDL VARCHAR(1680) NOT NULL,
 TYPE_OF_LOAD CHARACTER(1) NOT NULL,
 DELETERULE CHARACTER(1),
 UPDATERULE CHARACTER(1)
)
 IN QREPITSO.QCITSO;


CREATE TABLE ITSOQ.IBMQREP_APPLYENQ
(
 LOCKNAME INTEGER
)
 IN QREPITSO.QCITSO1;

INSERT INTO ITSOQ.IBMQREP_APPLYPARMS
 (qmgr, monitor_limit, trace_limit, monitor_interval, prune_interval,
 autostop, logreuse, logstdout, arch_level, term, deadlock_retries)
 VALUES
 ('MQ59', 10080, 10080, 300000, 300, 'N', 'N', 'N', '0901', 'Y', 3);
COMMIT;
```

4.5.5 STEP SETSHQ5: Create replication queue maps

In this step we create the replication queue maps using the SQL generated by the Replication Center, as shown in Example 4-11 and Example 4-12.

Example 4-11 Create replication queue maps 1/2

```
--Beginning of script 1--   DatabaseDB2OS390 (D8G1) [WARNING***Please do not alter this
line]--
CONNECT TO D8G1 USER mwiman using mwiman;
INSERT INTO ITSOQ.IBMQREP_SENDQUEUES
 (pubqmapname, sendq, message_format, msg_content_type, state,
 error_action, heartbeat_interval, max_message_size, description,
 apply_alias, apply_schema, recvq, apply_server)
 VALUES
 ('QMAP_SRC_TO_TRG', 'QREP.SRC.TO.TRG.SENDQ', 'C', 'T', 'A', 'S',
 60, 64, '', 'DB8A', 'ITSOQ', 'QREP.SRC.TO.TRG.RECVQ', 'DB8A');
COMMIT;
```

Example 4-12 Create replication queue maps 2/2

```
--Beginning of script 2--   DatabaseDB2OS390 (DT11) [WARNING***Please do not alter this
line]--
CONNECT TO DB8A USER mwiman using mwiman;
INSERT INTO ITSOQ.IBMQREP_RECVQUEUES
 (repqmapname, recvq, sendq, adminq, capture_alias, capture_schema,
 num_apply_agents, memory_limit, state, description, capture_server)
 VALUES
 ('QMAP_SRC_TO_TRG', 'QREP.SRC.TO.TRG.RECVQ',
 'QREP.SRC.TO.TRG.SENDQ', 'QREP.SRC.ADMINQ', 'D8GG', 'ITSOQ', 16, 2,
 'A', '', 'DB8G');
COMMIT;
```

4.5.6 STEP SETSHQ6: Create Q subscriptions for the test tables

The subscriptions for the test tables are defined using the SQL generated by the Replication Center, as shown in Example 4-13 through Example 4-18 on page 129.

For each of the subscriptions defined, the column HAS_LOADPHASE has a value of 'I' (automatic load), STATE has a value of 'N' (new subscription), and CHANGED_COLS_ONLY has a value of 'Y' (changed columns only for conflict rule).

Example 4-13 Create the subscriptions on the primary server SC53 1/3

```
--Beginning of script 1--   DatabaseDB2OS390 (D8G1) [WARNING***Please do not alter this
line]--

CONNECT TO D8G1 USER mwiman using mwiman;


INSERT INTO ITSOQ.IBMQREP_SUBS
 (subname, source_owner, source_name, sendq, subtype,
 all_changed_rows, before_values, changed_cols_only, has_loadphase,
 state, source_node, target_node, options_flag, suppress_deletes,
 target_server, target_alias, target_owner, target_name, target_type,
 apply_schema)
 VALUES
 ('PRODUCTS0001', 'ITSOQ', 'PRODUCTS', 'QREP.SRC.TO.TRG.SENDQ', 'U',
 'N', 'N', 'Y', 'I', 'N', 0, 0, 'NNNN', 'N', 'DB8A', 'DB8A', 'ITSOQ',
 'PRODUCTS', 1, 'ITSOQ');

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('PRODUCTS0001', 'PRODUCT_ID', 1);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('PRODUCTS0001', 'PRODUCT_NAME', 0);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('PRODUCTS0001', 'PRODUCT_CATEGORY', 0);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('PRODUCTS0001', 'PRICE', 0);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('PRODUCTS0001', 'MANUFACT_ID', 0);

-- COMMIT;
```

Example 4-14 Create the subscriptions on the primary server SC53 2/3

```
--Beginning of script 1--   DatabaseDB2OS390 (D8G1) [WARNING***Please do not alter this
line]-

CONNECT TO D8G1 USER mwiman using mwiman;


INSERT INTO ITSOQ.IBMQREP_SUBS
 (subname, source_owner, source_name, sendq, search_condition,
 subtype, all_changed_rows, before_values, changed_cols_only,
 has_loadphase, state, source_node, target_node, options_flag,
 suppress_deletes, target_server, target_alias, target_owner,
 target_name, target_type, apply_schema)
 VALUES
 ('SALES0001', 'ITSOQ', 'SALES', 'QREP.SRC.TO.TRG.SENDQ',
 'WHERE :REGION_CODE = ''EAST''', 'U', 'Y', 'N', 'Y', 'I', 'N', 0, 0,
 'NNNN', 'N', 'DB8A', 'DB8A', 'ITSOQ', 'SALES_EAST', 1, 'ITSOQ');

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0001', 'PRODUCT_ID', 1);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0001', 'STORE_ID', 2);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0001', 'QUANTITY_SOLD', 0);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0001', 'DATE_OF_SALE', 3);

-- COMMIT;
```

Example 4-15 Create the subscriptions on the primary server SC53 3/3

```
--Beginning of script 1--   DatabaseDB2OS390 (D8G1) [WARNING***Please do not alter this
line]--

CONNECT TO D8G1 USER mwiman using mwiman;


INSERT INTO ITSOQ.IBMQREP_SUBS
 (subname, source_owner, source_name, sendq, search_condition,
 subtype, all_changed_rows, before_values, changed_cols_only,
 has_loadphase, state, source_node, target_node, options_flag,
 suppress_deletes, target_server, target_alias, target_owner,
 target_name, target_type, apply_schema)
 VALUES
 ('SALES0002', 'ITSOQ', 'SALES', 'QREP.SRC.TO.TRG.SENDQ',
 'WHERE :REGION_CODE = ''WEST''', 'U', 'Y', 'N', 'Y', 'I', 'N', 0, 0,
 'NNNN', 'N', 'DB8A', 'DB8A', 'ITSOQ', 'SALES_WEST', 1, 'ITSOQ');

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
```

```
 (subname, src_colname, is_key)
 VALUES
 ('SALES0002', 'PRODUCT_ID', 1);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0002', 'STORE_ID', 2);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0002', 'QUANTITY_SOLD', 0);

INSERT INTO ITSOQ.IBMQREP_SRC_COLS
 (subname, src_colname, is_key)
 VALUES
 ('SALES0002', 'DATE_OF_SALE', 3);

-- COMMIT;
```

Example 4-16 Create the subscriptions on the secondary server SC539 1/3

```
--Beginning of script 2--   DatabaseDB2OS390 (DB8A) [WARNING***Please do not alter this
line]--

CONNECT TO DB8A USER mwiman using mwiman;


INSERT INTO ITSOQ.IBMQREP_TARGETS
 (subname, recvq, source_owner, source_name, target_owner,
 target_name, modelq, source_server, source_alias, target_type, state,
 subtype, conflict_rule, conflict_action, error_action, source_node,
 target_node, load_type, has_loadphase)
 VALUES
 ('PRODUCTS0001', 'QREP.SRC.TO.TRG.RECVQ', 'ITSOQ', 'PRODUCTS',
 'ITSOQ', 'PRODUCTS', 'IBMQREP.SPILL.MODELQ', 'DB8G', 'D8GG', 1, 'I',
 'U', 'K', 'S', 'S', 0, 0, 0, 'I');

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('PRODUCTS0001', 'QREP.SRC.TO.TRG.RECVQ', 'PRODUCT_ID', 'PRODUCT_ID'
, 'Y', 0);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('PRODUCTS0001', 'QREP.SRC.TO.TRG.RECVQ', 'PRODUCT_NAME',
 'PRODUCT_NAME', 'N', 1);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('PRODUCTS0001', 'QREP.SRC.TO.TRG.RECVQ', 'PRODUCT_CATEGORY',
 'PRODUCT_CATEGORY', 'N', 2);
```

```
INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('PRODUCTS0001', 'QREP.SRC.TO.TRG.RECVQ', 'PRICE', 'PRICE', 'N', 3);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('PRODUCTS0001', 'QREP.SRC.TO.TRG.RECVQ', 'MANUFACT_ID',
 'MANUFACT_ID', 'N', 4);

-- COMMIT;
```

Example 4-17 Create the subscriptions on the secondary server SC539 2/3

```
--Beginning of script 2--   DatabaseDB2OS390 (DB8A) [WARNING***Please do not alter this
line]--

CONNECT TO DB8A USER mwiman using mwiman;


INSERT INTO ITSOQ.IBMQREP_TARGETS
 (subname, recvq, source_owner, source_name, target_owner,
 target_name, modelq, source_server, source_alias, target_type, state,
 subtype, conflict_rule, conflict_action, error_action, source_node,
 target_node, load_type, has_loadphase, search_condition)
 VALUES
 ('SALES0001', 'QREP.SRC.TO.TRG.RECVQ', 'ITSOQ', 'SALES', 'ITSOQ',
 'SALES_EAST', 'IBMQREP.SPILL.MODELQ', 'DB8G', 'D8GG', 1, 'I', 'U',
 'K', 'S', 'S', 0, 0, 0, 'I', 'WHERE :REGION_CODE = ''EAST''');

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0001', 'QREP.SRC.TO.TRG.RECVQ', 'PRODUCT_ID', 'PRODUCT_ID',
 'Y', 0);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0001', 'QREP.SRC.TO.TRG.RECVQ', 'STORE_ID', 'STORE_ID', 'Y',
 1);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0001', 'QREP.SRC.TO.TRG.RECVQ', 'QUANTITY_SOLD',
 'QUANTITY_SOLD', 'N', 2);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0001', 'QREP.SRC.TO.TRG.RECVQ', 'DATE_OF_SALE',
 'DATE_OF_SALE', 'Y', 3);
```

```
-- COMMIT;
```

Example 4-18 Create the subscriptions on the secondary server SC59 3/3

```
--Beginning of script 2--   DatabaseDB2OS390 (DB8A) [WARNING***Please do not alter this
line]--



CONNECT TO DB8A USER mwiman using mwiman;


INSERT INTO ITSOQ.IBMQREP_TARGETS
 (subname, recvq, source_owner, source_name, target_owner,
 target_name, modelq, source_server, source_alias, target_type, state,
 subtype, conflict_rule, conflict_action, error_action, source_node,
 target_node, load_type, has_loadphase, search_condition)
 VALUES
 ('SALES0002', 'QREP.SRC.TO.TRG.RECVQ', 'ITSOQ', 'SALES', 'ITSOQ',
 'SALES_WEST', 'IBMQREP.SPILL.MODELQ', 'DB8G', 'D8GG', 1, 'I', 'U',
 'K', 'S', 'S', 0, 0, 0, 'I', 'WHERE :REGION_CODE = ''WEST''');

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0002', 'QREP.SRC.TO.TRG.RECVQ', 'PRODUCT_ID', 'PRODUCT_ID',
 'Y', 0);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0002', 'QREP.SRC.TO.TRG.RECVQ', 'STORE_ID', 'STORE_ID', 'Y',
 1);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0002', 'QREP.SRC.TO.TRG.RECVQ', 'QUANTITY_SOLD',
 'QUANTITY_SOLD', 'N', 2);

INSERT INTO ITSOQ.IBMQREP_TRG_COLS
 (subname, recvq, target_colname, source_colname, is_key,
 target_colNo)
 VALUES
 ('SALES0002', 'QREP.SRC.TO.TRG.RECVQ', 'DATE_OF_SALE',
 'DATE_OF_SALE', 'Y', 3);

-- COMMIT;
```

4.5.7 STEP SETSHQ7: Set up WebSphere MQ on both servers

Example 4-19 shows the definition of the WebSphere MQ objects on the primary server SC53. Note the highlighted queue-sharing group name MQZG in the WebSphere MQ object names and its use instead of the queue manager name.

Example 4-20 shows the definition of the WebSphere MQ objects on the secondary server SC59. Note the highlighted queue-sharing group name MQZG in the WebSphere MQ object names and its use instead of the queue manager name.

Example 4-19 JCL to define WebSphere MQ objects on the primary server SC53

```
//ASNQDEF  JOB NOTIFY=&SYSUID,
//         MSGCLASS=H,MSGLEVEL=(1,1),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//ASNQMQD1 EXEC PGM=CSQUTIL,PARM='MQZ1'
//STEPLIB   DD DSN=MQZ1.USERAUTH,DISP=SHR
//          DD DSN=MQ600.SCSQANLE,DISP=SHR
//          DD DSN=MQ600.SCSQAUTH,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *

*
* Following are the definitions which are required to send data
* from the source capture instance to the target apply instance.
* Source consists of shared queues between QMGR MQZ1 on SC53 and
* MQZ2 on SC67
*

DEFINE REPLACE                                                  +
       QLOCAL(QREP.SRC.ADMINQ)                                  +
       QSGDISP(SHARED)                                          +
       CFSTRUCT(APPLICATION1)                                   +
       DESCR('LOCAL DEFN OF ADMINQ FOR SOURCE CAPTURE')         +
       PUT(ENABLED)                                             +
       GET(ENABLED)                                             +
       SHARE                                                    +
       DEFSOPT(SHARED)                                          +
       DEFPSIST(YES)

DEFINE REPLACE                                                  +
       QLOCAL(QREP.SRC.RESTARTQ)                                +
       QSGDISP(SHARED)                                          +
       CFSTRUCT(APPLICATION1)                                   +
       DESCR('LOCAL DEFN OF RESTART FOR SOURCE CAPTURE')        +
       PUT(ENABLED)                                             +
       GET(ENABLED)                                             +
       SHARE                                                    +
       DEFSOPT(SHARED)                                          +
       DEFPSIST(YES)                                            +
       INDXTYPE(CORRELID)

DEFINE REPLACE                                                  +
       QREMOTE(QREP.SRC.TO.TRG.SENDQ)                           +
       QSGDISP(GROUP)                                           +
       DESCR('REMOTE DEFN OF SEND QUEUE FROM SOURCE TO TARGET') +
       PUT(ENABLED)                                             +
       XMITQ(MQTXMIT)                                           +
       RNAME(QREP.SRC.TO.TRG.RECVQ)                             +
       RQMNAME(MQ59)                                            +
       DEFPSIST(YES)
```

```
* Following are the definitions which are required to send data
* from an MQSeries queue manager named MQZ1 to a Queue manager
* named MQ59.  These definitions are required only once.
*

DEFINE REPLACE                                                  +
       QLOCAL(MQTXMIT)                                          +
       QSGDISP(SHARED)                                          +
       CFSTRUCT(APPLICATION1)                                   +
       DESCR('TRANSMISSION QUEUE TO MQ59')                      +
       USAGE(XMITQ)                                             +
       PUT(ENABLED)                                             +
       GET(ENABLED)                                             +
       TRIGGER                                                  +
       TRIGTYPE(FIRST)                                          +
       TRIGDATA(MQZG.TO.MQ59)                                   +
       INITQ(SYSTEM.CHANNEL.INITQ)

*
* Note that a listener must be started, on the other queue
* manager, for the port number identified in the channel
* sender definition.
*

DEFINE REPLACE                                                  +
       CHANNEL(MQZG.TO.MQ59)                                    +
       QSGDISP(GROUP)                                           +
       CHLTYPE(SDR)                                             +
       TRPTYPE(TCP)                                             +
       DESCR('SENDER CHANNEL TO MQ59')                          +
       XMITQ(MQTXMIT)                                           +
       CONNAME('9.12.4.10(1540)')

DEFINE REPLACE                                                  +
       CHANNEL(MQ59.TO.MQZG)                                    +
       QSGDISP(GROUP)                                           +
       CHLTYPE(RCVR)                                            +
       TRPTYPE(TCP)                                             +
       DESCR('RECEIVER CHANNEL FROM MQ59')

DISPLAY QUEUE(*) ALL
```

Example 4-20 JCL to define WebSphere MQ objects on the secondary server SC59

```
//ASNQDEF JOB (POK,999),HUEYKIM,MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M
//*
//* This is an example of MQSeries queue definitions for
//* Q Replication.
//*
//* Need to define ADMINQ, RESTARTQ, RECVQ and a model SPILLQ .
//* You might also need to define a Dead letter Q.
//*
//* Locate and change all occurrences of the following strings
//*  .  MQS!!0 to the name of your MQSeries target library
//*
//* The STEPLIB must include the MQSeries libraries
//* if they are not installed in the LNKLST.
//*
```

```
//*
//*******************************************************************/
//*
//ASNQMQD1 EXEC PGM=CSQUTIL,PARM='MQ59'
//STEPLIB   DD DSN=MQ59.USERAUTH,DISP=SHR
//          DD DSN=MQ600.SCSQANLE,DISP=SHR
//          DD DSN=MQ600.SCSQAUTH,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *

* Following are the definitions which are required to send data
* from an MQSeries queue manager named MQ59 to a Queue manager
* named MQZG.
*

DEFINE REPLACE                                                   +
       QLOCAL(MQSXMIT)                                           +
       DESCR('TRANSMISSION QUEUE TO MQZG')                       +
       USAGE(XMITQ)                                              +
       PUT(ENABLED)                                              +
       GET(ENABLED)                                              +
       TRIGGER                                                   +
       TRIGTYPE(FIRST)                                           +
       TRIGDATA(MQ59.TO.MQZG)                                    +
       INITQ(SYSTEM.CHANNEL.INITQ)

*
* Note that a listener must be started, on the other queue
* manager, for the port number identified in the channel
* sender definition. 9.12.6.11 is a DVIPA for both
* MQZ1 and MQZ2
*

DEFINE REPLACE                                                   +
       CHANNEL(MQ59.TO.MQZG)                                     +
       CHLTYPE(SDR)                                              +
       TRPTYPE(TCP)                                              +
       DESCR('SENDER CHANNEL TO MQZG')                           +
       XMITQ(MQSXMIT)                                            +
       CONNAME('9.12.6.11(1540)')

DEFINE REPLACE                                                   +
       CHANNEL(MQZG.TO.MQ59)                                     +
       CHLTYPE(RCVR)                                             +
       TRPTYPE(TCP)                                              +
       DESCR('RECEIVER CHANNEL FROM MQZG')

*
* Following are the definitions which are required to receive data
* at the target apply instance from the source capture instance.
*

DEFINE REPLACE                                                   +
       QMODEL('IBMQREP.SPILL.MODELQ')                            +
       DEFSOPT(SHARED)                                           +
       MAXDEPTH(500000)                                          +
       MSGDLVSQ(FIFO)                                            +
       DEFTYPE(PERMDYN)

DEFINE REPLACE                                                   +
       QREMOTE(QREP.SRC.ADMINQ)                                  +
       DESCR('REMOTE DEFN OF ADMINQ FOR SOURCE CAPTURE')         +
       PUT(ENABLED)                                              +
       XMITQ(MQSXMIT)                                            +
       RNAME(QREP.SRC.ADMINQ)                                    +
       RQMNAME(MQZG)                                             +
       DEFPSIST(YES)

DEFINE REPLACE                                                   +
       QLOCAL(QREP.SRC.TO.TRG.RECVQ)                             +
       DESCR('LOCAL RECEIVE QUEUE - SOURCE TO TARGET')           +
       PUT(ENABLED)                                              +
       GET(ENABLED)                                              +
       SHARE                                                     +
       DEFSOPT(SHARED)                                           +
       DEFPSIST(YES)                                             +
       INDXTYPE(MSGID)


DISPLAY QUEUE(*) ALL
```

4.5.8 STEP SETSHQ8: Start Q Capture and Q Apply on appropriate servers

Start Q Capture on the primary server SC53, as shown in Example 4-21, and Q Apply on the secondary server, as shown in Example 4-22, after ensuring that the channels are running and the transmit and receive queues are empty. Their successful initialization can be determined by viewing the contents of their respective logs, as described in Example 3-16 on page 50 and Example 3-12 on page 48.

Note: Since the subscriptions were defined with column STATE value of ‘N’ in the IBMQREP_SUBS table, the subscriptions are activated when Q Capture starts. The HAS_LOADPHASE value of ‘I’ causes an automatic load to be initiated. You can verify that replication is operating normally, as described in “STEP SWUFOPA17: Verify that Q replication up and running” on page 70.

Example 4-21 JCL to start Q Capture on the primary server SC53

```
//QCAPSHRQ JOB NOTIFY=&SYSUID,                                     JOB17436
//         MSGCLASS=H,MSGLEVEL=(1,0),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//QCAP     EXEC PGM=ASNQCAP,
// PARM='/CAPTURE_SERVER=D8GG capture_schema=ITSOQ startmode=warmsi
//             CAPTURE_PATH=//''HUEYKIM'
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8G8.SDSNLOAD,DISP=SHR
//         DD DSN=MQ600.SCSQANLE,DISP=SHR
//         DD DSN=MQ600.SCSQLOAD,DISP=SHR
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(50,70)),
//             DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
```

```
//SYSUDUMP DD  DUMMY
//*
```

Example 4-22 Start Q Apply on the secondary server SC59

```
//QAPPSHRQ JOB (POK,999),HUEYKIM,MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M,TIME=NOLIMIT
//QAPP EXEC PGM=ASNQAPP,
//  PARM='/APPLY_SERVER=DB8A APPLY_PATH=//''HUEYKIM
//             APPLY_SCHEMA=ITSOQ'
//*
//STEPLIB  DD DSN=ASN.V9R1M1.SASNLOAD,DISP=SHR
//         DD DSN=DB8A8.SDSNLOAD,DISP=SHR
//MSGS     DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//*
```

4.6 Test cases

To verify the seamless and synergistic relationship between WebSphere MQ shared queues and Q replication, we executed the following test cases in this configuration:

- Failover operations with no inflight automatic load in progress at the time of failure where unidirectional Q replication is restored to the secondary server SC59 without any data loss after the failure of LPAR SC53. The Q Capture process that was running on the failed LPAR SC53 is started (manually or automatically) on LPAR SC67.
- Failover operations with inflight automatic load in progress at the time of failure where unidirectional Q replication is restored to the secondary server SC59 without any data loss after the failure of LPAR SC53. The Q Capture process that was running on the failed LPAR SC53 is started (manually or automatically) on LPAR SC67.

These test cases and results are described in detail in the following sections.

Attention: Even though WebSphere enables access to shared queues from multiple LPARs, Q replication accesses these shared queues from only one LPAR or the other, in other words, mutually exclusive access to shared Q replication WebSphere MQ objects. Any attempt to start Q Capture on LPAR SC67 when it is already running on LPAR SC53 in our scenario results in an error message `ASN0554E “Q Capture” : “ITSOQ” : “Initial” : The program is already active`, as shown in the MVS™ log output in Example 4-23.

Example 4-23 MVS log output showing ASN0554E error message

```
$HASP100 QCAPSHRX ON INTRDR                            FROM TSU02673
HUEYKIM
IRR010I  USERID HUEYKIM  IS ASSIGNED TO THIS JOB.
ICH70001I HUEYKIM  LAST ACCESS AT 21:33:31 ON FRIDAY, MARCH 3, 2006
$HASP373 QCAPSHRX STARTED - INIT A    - CLASS A - SYS SC67
IEF403I QCAPSHRX - STARTED - TIME=21.43.23 - ASID=0025 - SC67
$CJ(3024),P
$HASP890 JOB(QCAPSHRX) 085
$HASP890 JOB(QCAPSHRX)  STATUS=(AWAITING PURGE),CLASS=A,
$HASP890                PRIORITY=1,SYSAFF=(ANY),
$HASP890                HOLD=(NONE),PURGE=YES,CANCEL=YES
$HASP250 QCAPSHRX PURGED -- (JOB KEY WAS BE73288B)
```

```
ASN0544E  "Q Capture" : "ITSOQ" : "Initial" : The program is already
active.
ASN0573I  "Q Capture" : "ITSOQ" : "Initial" : The program was stopped.
DSNT376I  -D8G2 PLAN=ASNQC910 WITH 089
        CORRELATION-ID=QCAPSHRXHold
        CONNECTION-ID=RRSAF
        LUW-ID=USIBMSC.SCPD8G2.BE7328ACBC1B=893
        THREAD-INFO=HUEYKIM:*:HoldLThread:ITSOQ
        IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=ASNQC910
WITH
        CORRELATION-ID=QCAPSHRQHold
        CONNECTION-ID=RRSAF
        LUW-ID=USIBMSC.SCPD8G1.BE732848F8C4=28
        THREAD-INFO=HUEYKIM:*:HoldLThread:ITSOQ
        ON MEMBER D8G1
DSNT501I  -D8G2 DSNILMCL RESOURCE UNAVAILABLE 090
           CORRELATION-ID=QCAPSHRXHold
           CONNECTION-ID=RRSAF
           LUW-ID=USIBMSC.SCPD8G2.BE7328ACBC1B=351127
           REASON 00C9008E
           TYPE 00000D01
           NAME 00000283.00000370
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QCAPSHRX          QCAP        12   2170    .00    .00    1.1   2600
 0      0      0      0     0
IEF404I QCAPSHRX - ENDED - TIME=21.44.34 - ASID=0025 - SC67
-QCAPSHRX ENDED.  NAME-                     TOTAL CPU TIME=   .00
TOTAL ELAPSED TIME=   1.1
$HASP395 QCAPSHRX ENDED
```

4.6.1 Failover operations with no inflight automatic load in progress

The objective of this test case is to demonstrate that under failover operations with no inflight automatic load in progress, Q replication operates in a seamless fashion in our WebSphere MQ shared queues high availability primary server environment with no data loss.

The following assumptions are made about the test case environment:

- Application workloads are running against both members of the DB2 data sharing group D8GG comprising member databases D8G1 on LPAR SC53 and D8G2 on LPAR SC67.
- Q Capture is running on the primary server LPAR SC53.
- Q Apply is running on the secondary server LPAR SC59.
- The PRODUCTS table is replicated in its entirety to the secondary server.
- The source SALES table is replicated to a SALES_EAST and a SALES_WEST table on the target database depending on the REGION_CODE value on the source table. REGION_CODE column is not replicated.

Figure 4-5 shows the steps executed in this test case.

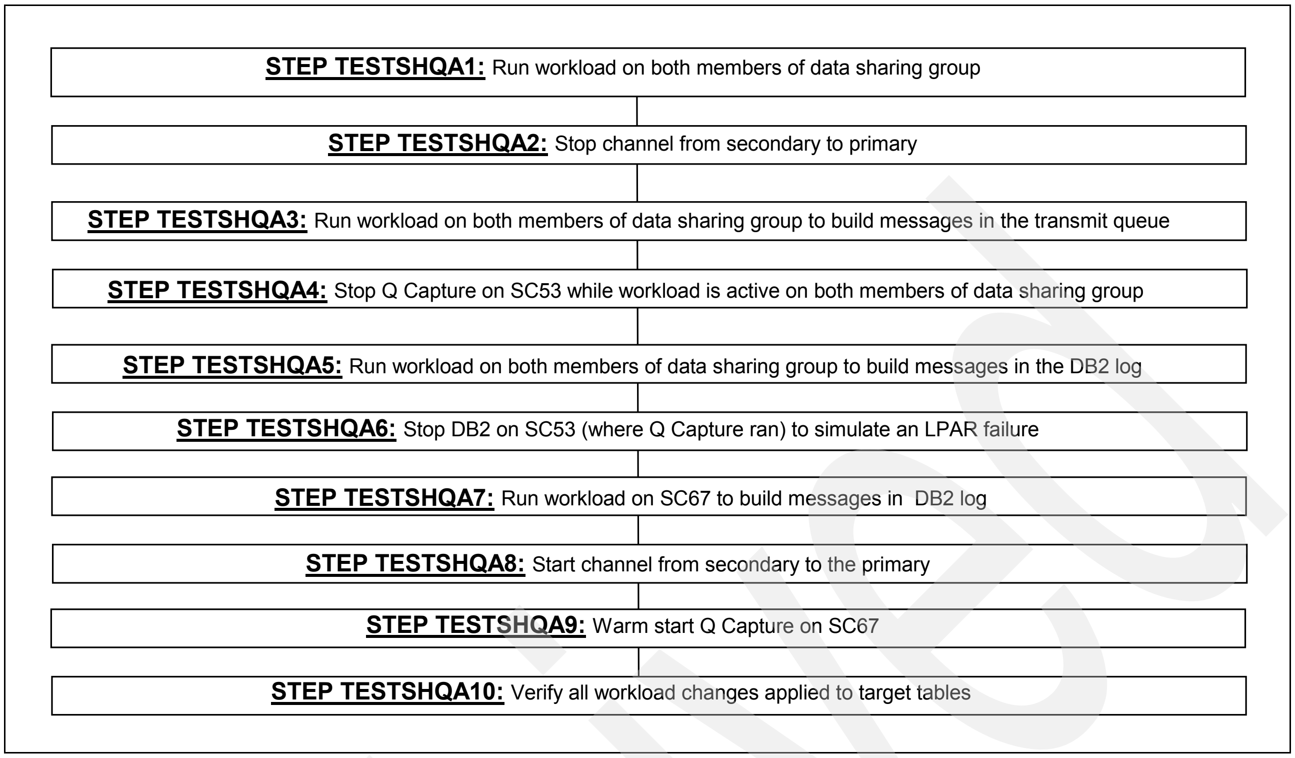

STEP TESTSHQA1: Run workload on both members of data sharing group
STEP TESTSHQA2: Stop channel from secondary to primary
STEP TESTSHQA3: Run workload on both members of data sharing group to build messages in the transmit queue
STEP TESTSHQA4: Stop Q Capture on SC53 while workload is active on both members of data sharing group
STEP TESTSHQA5: Run workload on both members of data sharing group to build messages in the DB2 log
STEP TESTSHQA6: Stop DB2 on SC53 (where Q Capture ran) to simulate an LPAR failure
STEP TESTSHQA7: Run workload on SC67 to build messages in DB2 log
STEP TESTSHQA8: Start channel from secondary to the primary
STEP TESTSHQA9: Warm start Q Capture on SC67
STEP TESTSHQA10: Verify all workload changes applied to target tables

Figure 4-5 Failover operations with no inflight automatic load in progress test case with shared queues

Each of the steps shown in Figure 4-5 is described in more detail here.

STEP TESTSHQA1: Run workload on all members of data sharing group

Example 4-24 shows the SQL used to simulate the workload running on both members of the data sharing group—database D8G1 on LPAR SC53 and D8G2 on LPAR SC67.

The contents of the tables on the primary server SC53 and secondary server are shown in Example 4-25 on page 138 and Example 4-26 on page 139, respectively. The content is identical on the primary server and secondary server, indicating successful replication with no data loss.

Example 4-24 Run workload on members D8G1 and D8G2 in the data sharing group D8GG

```
CONNECT TO D8G1 USER HUEYKIM USING

   Database Connection Information

 Database server        = DB2 OS/390 8.1.5

 SQL authorization ID   = HUEYKIM

 Local database alias   = D8G1

DELETE FROM ITSOQ.PRODUCTS

SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of a
query is an empty table.  SQLSTATE=02000

DELETE FROM ITSOQ.SALES

SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of a
```

```
query is an empty table.  SQLSTATE=02000

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (10, 'WIDGETS', 'TOOLS', 1.00, 100)

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (20, 'THINGAMABOBBERS', 'TOOLS', 5.00, 100)

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (30, 'WHATCHAMACALLITS', 'TOOLS', 10.00, 100)

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (40, 'DOODADS', 'TOOLS', 3.00, 100)

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (50, 'THINGAMAGIGS', 'TOOLS', 2.00, 100)

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (10, 100, 1000, '2006-03-01', 'EAST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (20, 120, 2000, '2006-03-01', 'EAST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (30, 100, 5000, '2006-03-02', 'EAST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (40, 140, 4000, '2006-03-02', 'EAST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (50, 150, 5000, '2006-03-02', 'EAST')

DB20000I  The SQL command completed successfully.

CONNECT RESET

DB20000I  The SQL command completed successfully.

------------------------------------------------------------------------------------
CONNECT TO D8G2 USER HUEYKIM USING
Database Connection Information
Database server        = DB2 OS/390 8.1.5
```

```
SQL authorization ID   = HUEYKIM
Local database alias   = D8G2

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (30, 200, 5000, '2006-03-01', 'WEST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (40, 240, 4000, '2006-03-01', 'WEST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (50, 250, 5000, '2006-03-01', 'WEST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (10, 200, 1000, '2006-03-02', 'WEST')

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (20, 220, 2000, '2006-03-02', 'WEST')

DB20000I  The SQL command completed successfully.

CONNECT RESET

DB20000I  The SQL command completed successfully.

TERMINATE

DB20000I  The TERMINATE command completed successfully.
```

Example 4-25 Contents of tables in database D8G1 on the primary server SC53

```
connect to d8g1 user mwiman using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = MWIMAN
Local database alias   = D8G1

select * from itsoq.products order by product_id

PRODUCT_ID  PRODUCT_NAME         PRODUCT_CATEGORY PRICE             MANUFACT_ID
----------- -------------------- ---------------- ----------------- -----------
10           WIDGETS              TOOLS            1.00              100
20           THINGAMABOBBERS      TOOLS            5.00              100
30           WHATCHAMACALLITS     TOOLS           10.00              100
40           DOODADS              TOOLS            3.00              100
50           THINGAMAGIGS         TOOLS            2.00              100
5 record(s) selected.

select * from itsoq.sales order by region_code, product_id, store_id, date_of_sale

PRODUCT_ID  STORE_ID    QUANTITY_SOLD DATE_OF_SALE REGION_CODE
----------- ----------- ------------- ------------ -----------
10         100          1000              2006-03-01   EAST
```

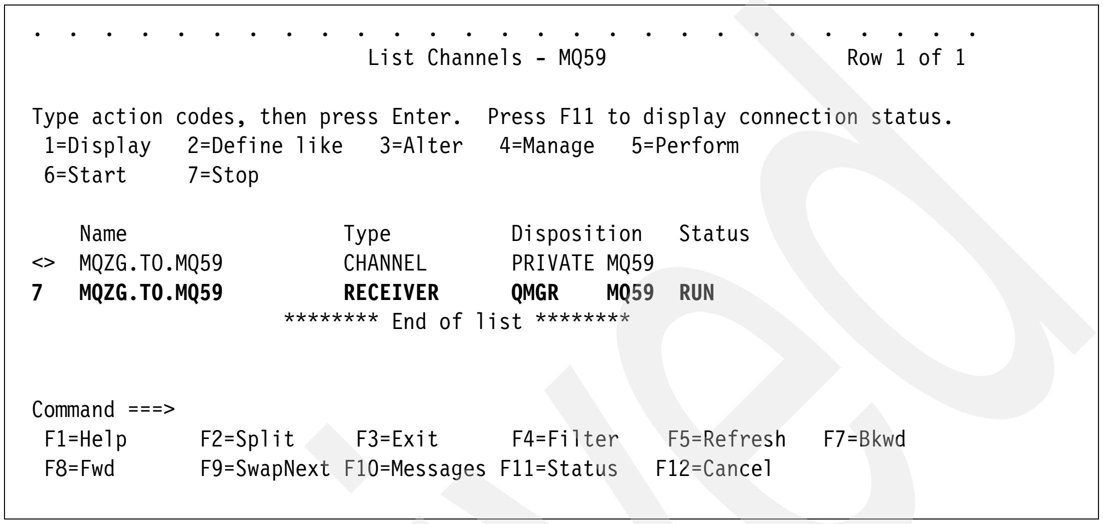
```
20          120         2000                 2006-03-01   EAST
30          100         5000                 2006-03-02   EAST
40          140         4000                 2006-03-02   EAST
50          150         5000                 2006-03-02   EAST
10          200         1000                 2006-03-02   WEST
20          220         2000                 2006-03-02   WEST
30          200         5000                 2006-03-01   WEST
40          240         4000                 2006-03-01   WEST
50          250         5000                 2006-03-01   WEST
10 record(s) selected.

terminate

DB20000I  The TERMINATE command completed successfully.
```

Example 4-26 Contents of tables in database DB8A on the secondary server SC59

```
connect to db8a user mwiman using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = MWIMAN
Local database alias   = DB8A

select * from itsoq.products order by product_id

PRODUCT_ID PRODUCT_NAME         PRODUCT_CATEGORY        PRICE          MANUFACT_ID
----------- -------------------- ---------------- ----------------- -----------
10           WIDGETS            TOOLS                           1.00         100
20           THINGAMABOBBERS    TOOLS                           5.00         100
30           WHATCHAMACALLITS   TOOLS                          10.00         100
40           DOODADS            TOOLS                           3.00         100
50           THINGAMAGIGS       TOOLS                           2.00         100
5 record(s) selected.

select * from itsoq.sales_east order by product_id, store_id, date_of_sale

PRODUCT_ID  STORE_ID    QUANTITY_SOLD DATE_OF_SALE
----------- ----------- ------------- ------------
10            100          1000        2006-03-01
20            120          2000        2006-03-01
30            100          5000        2006-03-02
40            140          4000        2006-03-02
50            150          5000        2006-03-02
5 record(s) selected.

select * from itsoq.sales_west order by product_id, store_id, date_of_sale

PRODUCT_ID  STORE_ID    QUANTITY_SOLD DATE_OF_SALE
----------- ----------- ------------- ------------
10            200          1000        2006-03-02
20            220          2000        2006-03-02
30            200          5000        2006-03-01
40            240          4000        2006-03-01
50            250          5000        2006-03-01
5 record(s) selected.

terminate

DB20000I  The TERMINATE command completed successfully.
```

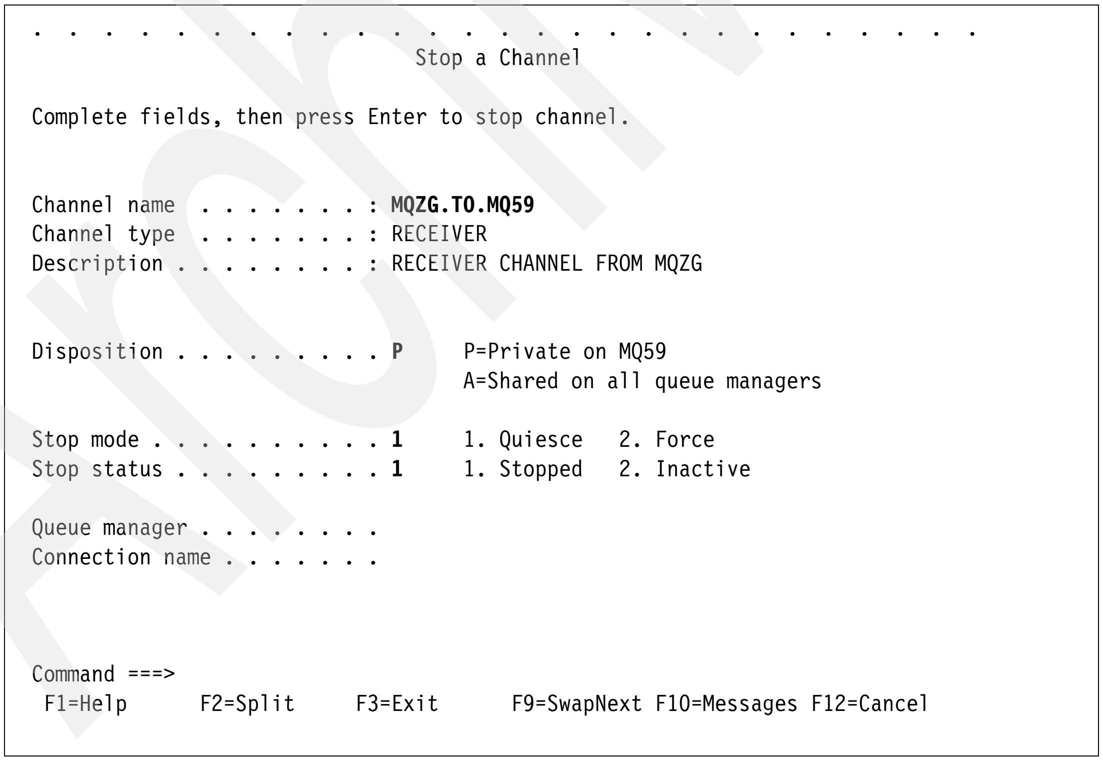
STEP TESTSHQA2: Stop channel from secondary to primary

In this step, stop the channel between the secondary server SC59 and the primary server SC53, as shown in Figure 4-6 through Figure 4-9, in order to be able to build changes in the transmit queue in the coupling facility for processing by LPAR SC67 when Q Capture is started on that LPAR.



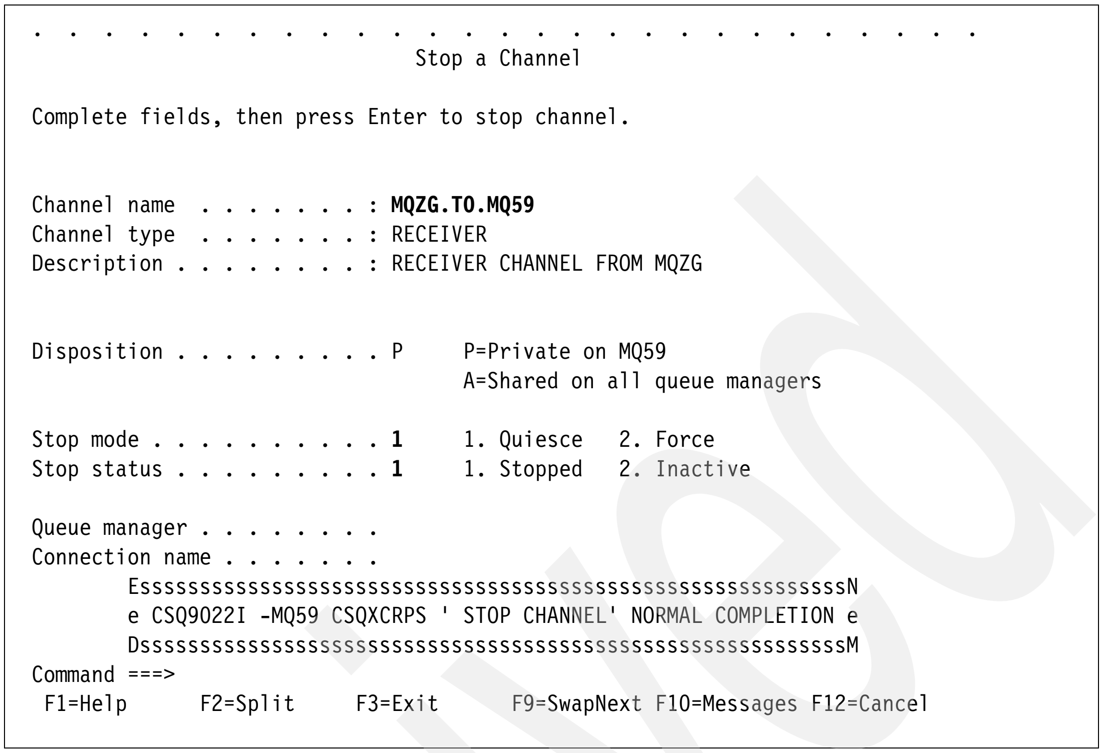
```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                            List Channels - MQ59                    Row 1 of 1

 Type action codes, then press Enter.  Press F11 to display connection status.
  1=Display   2=Define like   3=Alter   4=Manage   5=Perform
  6=Start     7=Stop

     Name                  Type          Disposition   Status
 <>  MQZG.TO.MQ59          CHANNEL       PRIVATE MQ59
 7   MQZG.TO.MQ59          RECEIVER      QMGR    MQ59  RUN
                     ******** End of list ********


 Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F7=Bkwd
  F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 4-6 Stop channel from the secondary server SC59 to the primary server SC53 1/4



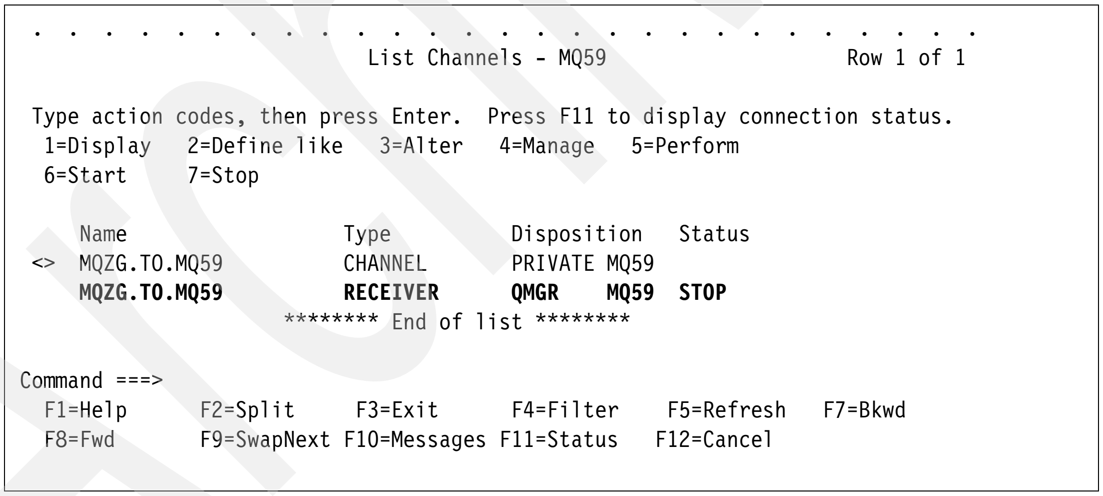
```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                                Stop a Channel

 Complete fields, then press Enter to stop channel.


 Channel name  . . . . . . . : MQZG.TO.MQ59
 Channel type  . . . . . . . : RECEIVER
 Description . . . . . . . . : RECEIVER CHANNEL FROM MQZG


 Disposition . . . . . . . . . P     P=Private on MQ59
                                     A=Shared on all queue managers

 Stop mode . . . . . . . . . . 1     1. Quiesce   2. Force
 Stop status . . . . . . . . . 1     1. Stopped   2. Inactive

 Queue manager . . . . . . . .
 Connection name . . . . . . .



 Command ===>
  F1=Help      F2=Split     F3=Exit      F9=SwapNext F10=Messages F12=Cancel
```

Figure 4-7 Stop channel from the secondary server SC59 to the primary server SC53 2/4



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                                Stop a Channel

 Complete fields, then press Enter to stop channel.


 Channel name  . . . . . . . : MQZG.TO.MQ59
 Channel type  . . . . . . . : RECEIVER
 Description . . . . . . . . : RECEIVER CHANNEL FROM MQZG


 Disposition . . . . . . . . . P     P=Private on MQ59
                                     A=Shared on all queue managers

 Stop mode . . . . . . . . . . 1     1. Quiesce   2. Force
 Stop status . . . . . . . . . 1     1. Stopped   2. Inactive

 Queue manager . . . . . . . .
 Connection name . . . . . . .
         EssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssN
         e CSQ9022I -MQ59 CSQXCRPS ' STOP CHANNEL' NORMAL COMPLETION e
         DssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssM
 Command ===>
  F1=Help      F2=Split     F3=Exit      F9=SwapNext F10=Messages F12=Cancel
```

Figure 4-8 Stop channel from the secondary server SC59 to the primary server SC53 3/4



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                            List Channels - MQ59                  Row 1 of 1

 Type action codes, then press Enter.  Press F11 to display connection status.
  1=Display   2=Define like   3=Alter   4=Manage   5=Perform
  6=Start     7=Stop

     Name                  Type          Disposition   Status
 <>  MQZG.TO.MQ59          CHANNEL       PRIVATE MQ59
     MQZG.TO.MQ59          RECEIVER      QMGR    MQ59  STOP
                     ******** End of list ********

Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F7=Bkwd
  F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

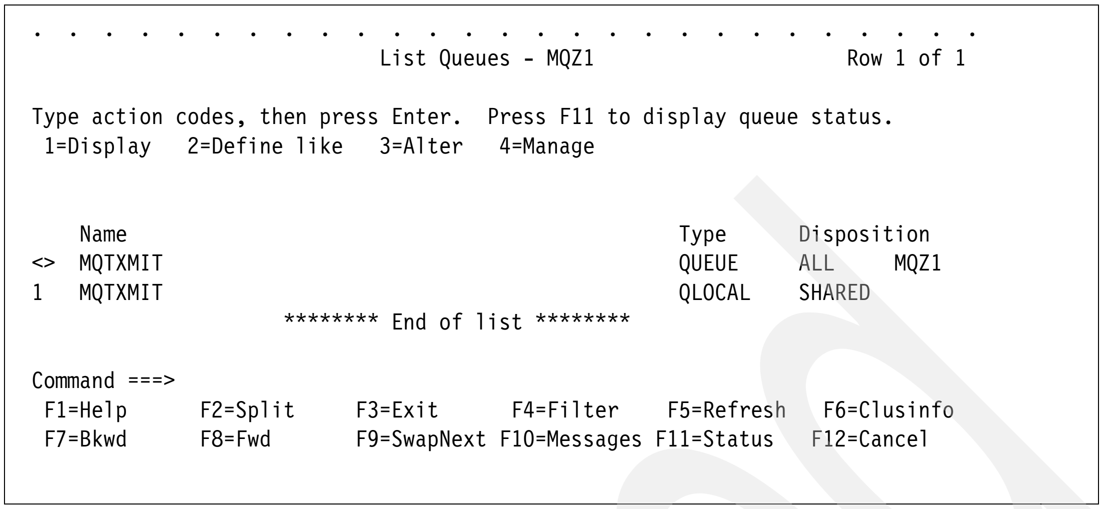
Figure 4-9 Stop channel from the secondary server SC59 to the primary server SC53 4/4

STEP TESTSHQA3: Run workload on all members of data sharing group

In this step, run another workload, as shown in Example 4-27, on both members of the data sharing group to generate messages in the transmit queue in the coupling facility. Figure 4-10 on page 143 and Figure 4-11 on page 143 show the build up of messages (current queue depth of 11) in the transmit queue in the coupling facility.

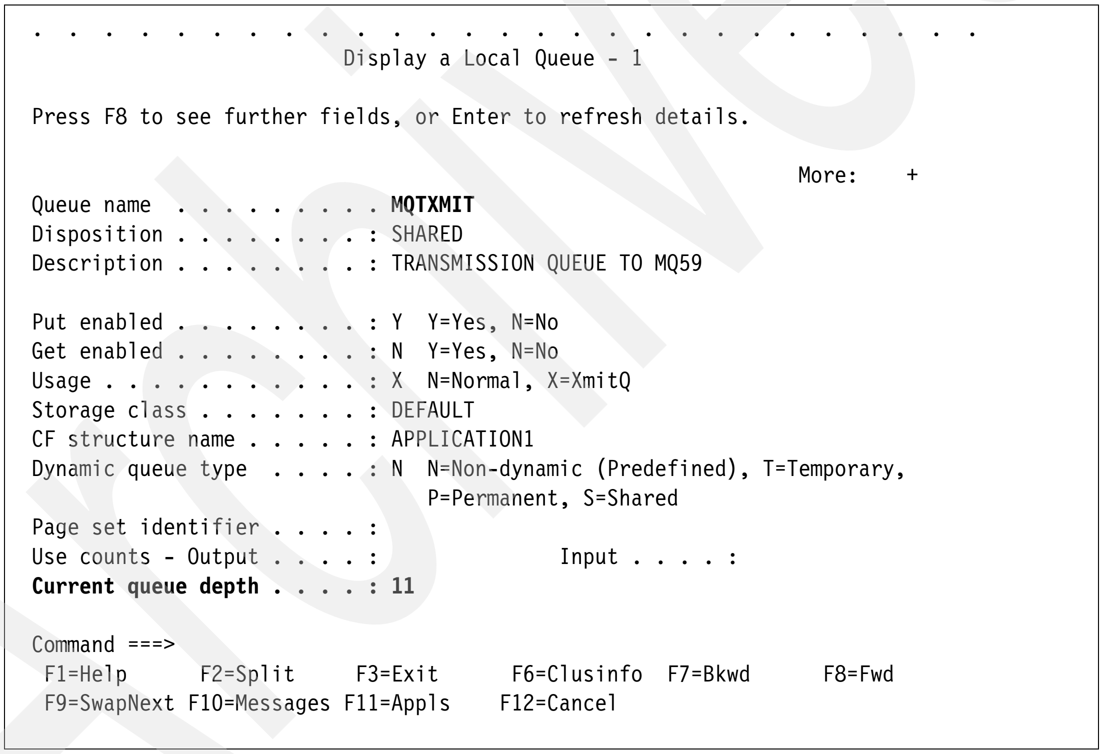
Example 4-27 Run another workload on the members D8G1 and D8G2 in the data-sharing group D8GG

```
CONNECT TO D8G1 USER HUEYKIM USING
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = HUEYKIM
Local database alias   = D8G1

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (60, 'HEAVY HAMMER', 'TOOLS', 100.00, 100)

DB20000I  The SQL command completed successfully.

UPDATE ITSOQ.PRODUCTS SET PRICE = 8.00 WHERE PRODUCT_ID = 20

DB20000I  The SQL command completed successfully.

DELETE FROM ITSOQ.PRODUCTS WHERE PRODUCT_ID = 50

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (60, 160, 6000, '2006-03-03', 'EAST')

DB20000I  The SQL command completed successfully.

DELETE FROM ITSOQ.SALES WHERE PRODUCT_ID = 50

DB20000I  The SQL command completed successfully.

CONNECT RESET

DB20000I  The SQL command completed successfully.

CONNECT TO D8G2 USER HUEYKIM USING

Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = HUEYKIM
Local database alias   = D8G2

UPDATE ITSOQ.PRODUCTS  SET MANUFACT_ID = 200 WHERE PRODUCT_ID = 20

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (30, 200, 3000, '2006-03-03', 'WEST')

DB20000I  The SQL command completed successfully.

UPDATE ITSOQ.SALES SET REGION_CODE = 'WEST' WHERE PRODUCT_ID = 40 AND STORE_ID = 140 AND
DATE_OF_SALE = '2006-03-02'

DB20000I  The SQL command completed successfully.

CONNECT RESET

DB20000I  The SQL command completed successfully.

TERMINATE

DB20000I  The TERMINATE command completed successfully.
```



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                             List Queues - MQZ1                    Row 1 of 1

 Type action codes, then press Enter.  Press F11 to display queue status.
  1=Display   2=Define like   3=Alter   4=Manage


     Name                                        Type      Disposition
 <>  MQTXMIT                                     QUEUE     ALL     MQZ1
 1   MQTXMIT                                     QLOCAL    SHARED
                      ******** End of list ********

 Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F6=Clusinfo
  F7=Bkwd      F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 4-10 View the depth of the transmit queue in the coupling facility 1/2



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                          Display a Local Queue - 1

 Press F8 to see further fields, or Enter to refresh details.

                                                                More:    +
 Queue name  . . . . . . . . MQTXMIT
 Disposition . . . . . . . . : SHARED
 Description . . . . . . . . : TRANSMISSION QUEUE TO MQ59

 Put enabled . . . . . . . . : Y  Y=Yes, N=No
 Get enabled . . . . . . . . : N  Y=Yes, N=No
 Usage . . . . . . . . . . . : X  N=Normal, X=XmitQ
 Storage class . . . . . . . : DEFAULT
 CF structure name . . . . . : APPLICATION1
 Dynamic queue type  . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                  P=Permanent, S=Shared
 Page set identifier . . . . :
 Use counts - Output . . . . :              Input . . . . :
 Current queue depth . . . . : 11

 Command ===>
  F1=Help      F2=Split     F3=Exit      F6=Clusinfo  F7=Bkwd      F8=Fwd
  F9=SwapNext F10=Messages F11=Appls    F12=Cancel
```

Figure 4-11 View the depth of the transmit queue in the coupling facility 2/2

STEP TESTSHQA4: Stop Q Capture on SC53 while workload is active

In this step, stop Q Capture on LPAR SC53, as shown in Example 4-28, in order to be able to build changes in the DB2 logs for processing by Q Capture when it is started on LPAR SC67. Its successful stopping can be determined by viewing the contents of the Q Capture log, similar to that shown in “Stop Q Capture log contents on the secondary server SC59” on page 43.

Example 4-28 Stop Q Capture on the primary server SC53

```
F QCAPSHRQ, STOP
ASN0522I  "Q Capture" : "ITSOQ" : "Initial" : The program received
```

```
the "STOP" command.
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-QCAPSHRQ         QCAP        00   8307   4.24    .00   13.3  2933K
 0      0      0      0     0
IEF404I QCAPSHRQ - ENDED - TIME=21.13.41 - ASID=0031 - SC53
-QCAPSHRQ ENDED.  NAME-                     TOTAL CPU TIME=  4.24
TOTAL ELAPSED TIME=  13.3
$HASP395 QCAPSHRQ ENDED
$HASP309 INIT A    INACTIVE ******** C=ABCDE
SE '21.13.41 JOB03002 $HASP165 QCAPSHRQ ENDED AT WTSCPLX1  MAXCC=0',
LOGON,USER=(HUEYKIM)
```

STEP TESTSHQA5: Run workload on all members of data sharing group

In this step, run another workload, as shown in Example 4-29, on both members of the data sharing group to generate messages in the DB2 log for Q Capture to process when it is started on LPAR SC67.

Example 4-29 Run workload on members D8G1 and D8G2 in the data sharing group D8GG

```
CONNECT TO D8G1 USER HUEYKIM USING
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = HUEYKIM
Local database alias   = D8G1

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (80, 'FLAMETHROWERS', 'TOOLS', 1000.00, 300)

DB20000I  The SQL command completed successfully.

INSERT INTO ITSOQ.SALES  (PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (80, 270, 5, '2006-03-04', 'WEST')

DB20000I  The SQL command completed successfully.

CONNECT RESET

DB20000I  The SQL command completed successfully.

CONNECT TO D8G2 USER HUEYKIM USING
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = HUEYKIM
Local database alias   = D8G2

UPDATE ITSOQ.PRODUCTS SET PRODUCT_CATEGORY = 'WEAPONS' WHERE PRODUCT_ID = 80

DB20000I  The SQL command completed successfully.

UPDATE ITSOQ.SALES SET QUANTITY_SOLD = 6000 WHERE PRODUCT_ID =40 AND STORE_ID = 240 AND
DATE_OF_SALE = '2006-03-01'

DB20000I  The SQL command completed successfully.

CONNECT RESET
```

```
DB20000I  The SQL command completed successfully.

TERMINATE

DB20000I  The TERMINATE command completed successfully.
```

STEP TESTSHQA6: Stop DB2 on SC53 (where Q Capture ran)

In this step, stop DB2 on LPAR SC53 to simulate its failure, as shown in Example 4-30.

Example 4-30   STOP DB2 MODE(FORCE)

```
-D8G1 STO DB2 MODE(FORCE)
DSNY002I  -D8G1 SUBSYSTEM STOPPING
DSNL005I  -D8G1 DDF IS STOPPING
DSNL006I  -D8G1 DDF STOP COMPLETE
ARC0723I BACKUP ENDING ON VOLUME TST025 AT 21:20:09, 986
ARC0723I (CONT.) 00000001 DATA SETS BACKED UP
CSQ5015I =MQZ1 CSQ5MONR Connection to D8G1 lost, DB2 shut down
forcibly
CSQ5003A =MQZ1 CSQ5CONN Connection to DB2 using D8GG 333
pending, no active DB2
+CSQX483E =MQZ1 CSQXSUPR DB2 not available
ATR169I RRS HAS UNSET EXITS FOR RESOURCE MANAGER 335
DSN.RRSATF.IBM.D8G1 REASON: UNREGISTERED
ATR169I RRS HAS UNSET EXITS FOR RESOURCE MANAGER 336
DSN.RRSPAS.IBM.D8G1 REASON: UNREGISTERED
IEC205I SYS15180,DFSMSHSM,HSMSC47,FILESEQ=00001, EXTEND VOLUME LIST,
987
DSN=TOTHSM.BACKTAPE.DATASET,VOLS=THS155
.............
..................
DSN9022I  -D8G1 DSNYASCP 'STO DB2' NORMAL COMPLETION
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
ARC0723I BACKUP ENDING ON VOLUME TOTDI1 AT 21:20:30, 022
ARC0723I (CONT.) 00000000 DATA SETS BACKED UP
-D8G1MSTR IEFPROC  D8G1MSTR    00  24260    .23    .09  402.2   225K
 0      0      0      0     0
IEF404I D8G1MSTR - ENDED - TIME=21.20.30 - ASID=0089 - SC53
-D8G1MSTR ENDED.  NAME-                     TOTAL CPU TIME=   .23
TOTAL ELAPSED TIME= 402.2
IEF352I ADDRESS SPACE UNAVAILABLE
$HASP395 D8G1MSTR ENDED
.................
.................
```

STEP TESTSHQA7: Run workload on SC67 to build messages in log

In this step, run another workload, as shown in Example 4-31, on LPAR SC67 only (since LPAR SC53 had failed) to generate messages in the DB2 log for Q Capture to process when it is started on LPAR SC67.

Example 4-31   Run workload on member D8G2 in the data sharing group D8GG

```
CONNECT TO D8G2 USER HUEYKIM USING
Database Connection Information
Database server        = DB2 OS/390 8.1.5
```

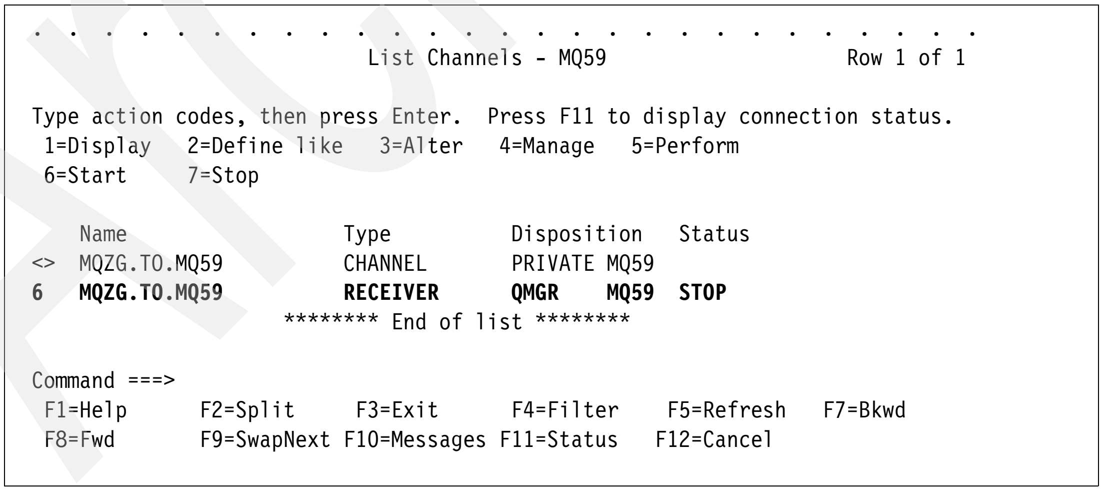
```
SQL authorization ID   = HUEYKIM
Local database alias   = D8G2

INSERT INTO ITSOQ.PRODUCTS  (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE,
MANUFACT_ID) VALUES (90, 'STRAWMEN', 'TOOLS', 8.00, 100)

DB20000I  The SQL command completed successfully.

UPDATE ITSOQ.SALES SET QUANTITY_SOLD = 0 WHERE PRODUCT_ID = 10 AND STORE_ID = 200 AND
DATE_OF_SALE = '2006-03-02'

DB20000I  The SQL command completed successfully.

CONNECT RESET

DB20000I  The SQL command completed successfully.

TERMINATE

DB20000I  The TERMINATE command completed successfully.
```

STEP TESTSHQA8: Start channel from secondary to primary

The status at this point is as follows:

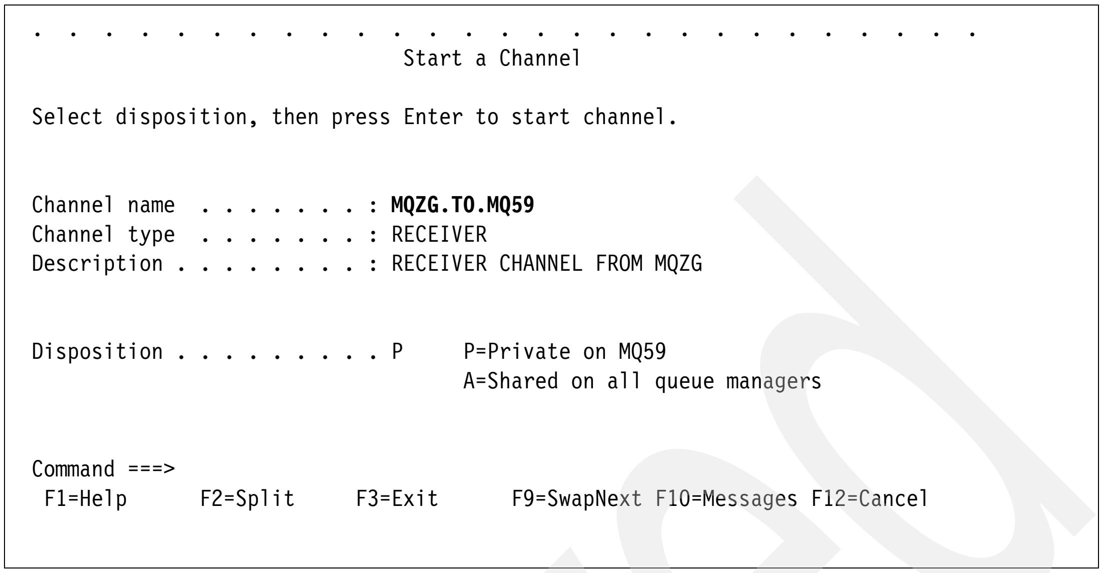
- Unpropagated changes exist on the DB2 log on database D8G1, DB2 log on database D8G2, and the transmit queue in the coupling facility.
- LPAR SC53 has crashed.
- The channel from the secondary server SC59 to the primary server SC53 is stopped.

Before Q Capture is started on LPAR SC67, start the channel as shown in Figure 4-12 through Figure 4-19 on page 149 to enable changes from the transmit queue in the coupling facility to propagate right away to the receive queue on the secondary server SC59. Figure 4-20 on page 150 shows the contents of the transmit queue being reduced to zero.



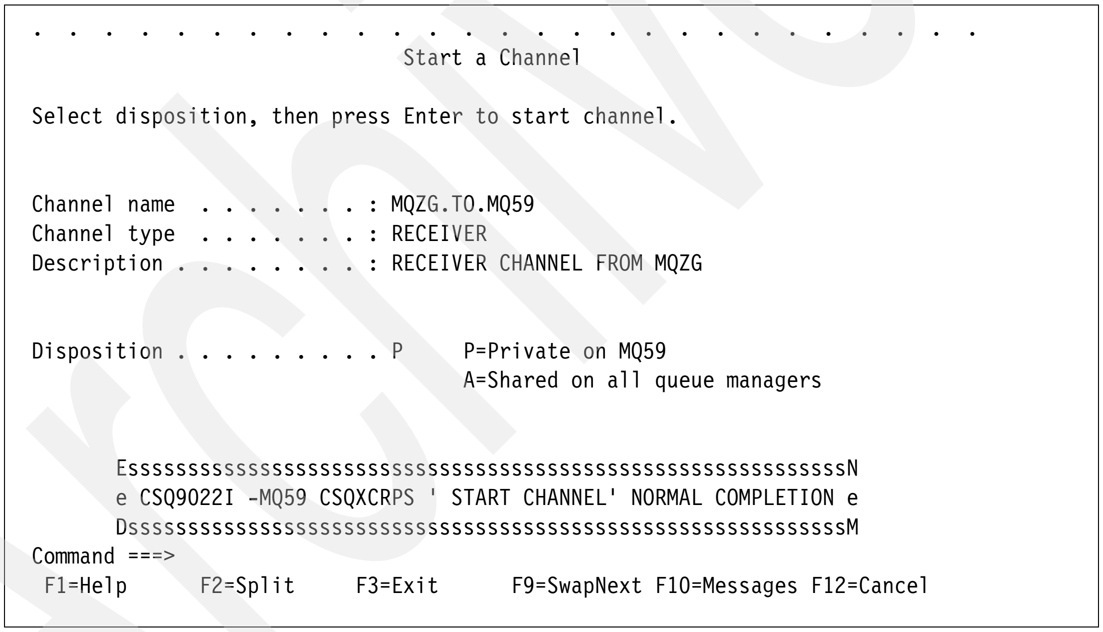
```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                            List Channels - MQ59                  Row 1 of 1

Type action codes, then press Enter.  Press F11 to display connection status.
 1=Display   2=Define like   3=Alter   4=Manage   5=Perform
 6=Start     7=Stop

    Name                  Type          Disposition   Status
<>  MQZG.TO.MQ59          CHANNEL       PRIVATE MQ59
6   MQZG.TO.MQ59          RECEIVER      QMGR    MQ59  STOP
                   ******** End of list ********

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F7=Bkwd
 F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 4-12   Start the channel between the secondary server SC59 & primary server SC53 1/9



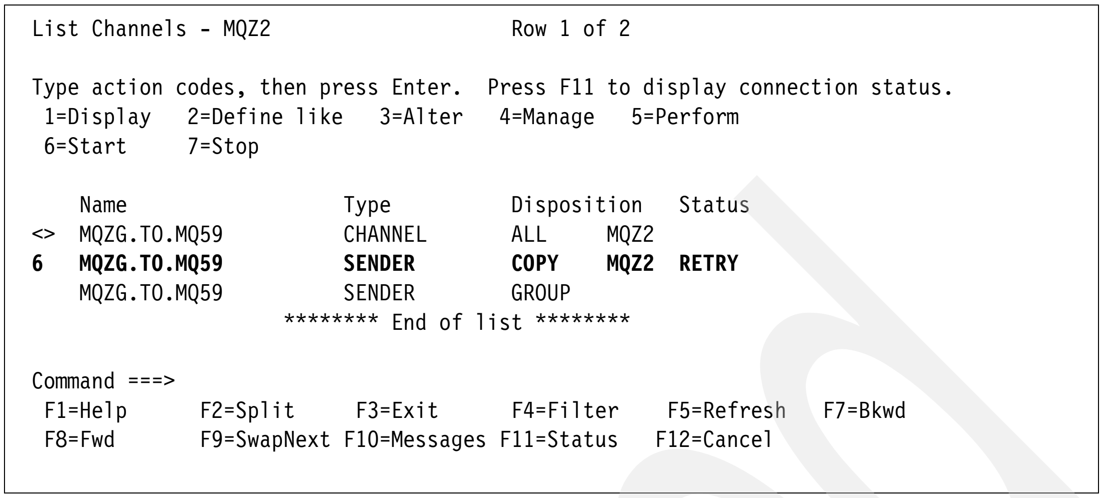
```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                               Start a Channel

 Select disposition, then press Enter to start channel.


 Channel name  . . . . . . . : **MQZG.TO.MQ59**
 Channel type  . . . . . . . : RECEIVER
 Description . . . . . . . . : RECEIVER CHANNEL FROM MQZG


 Disposition . . . . . . . . . P     P=Private on MQ59
                                     A=Shared on all queue managers



 Command ===>
  F1=Help      F2=Split     F3=Exit      F9=SwapNext F10=Messages F12=Cancel
```

Figure 4-13   Start the channel between the secondary server SC59 & primary server SC53 2/9



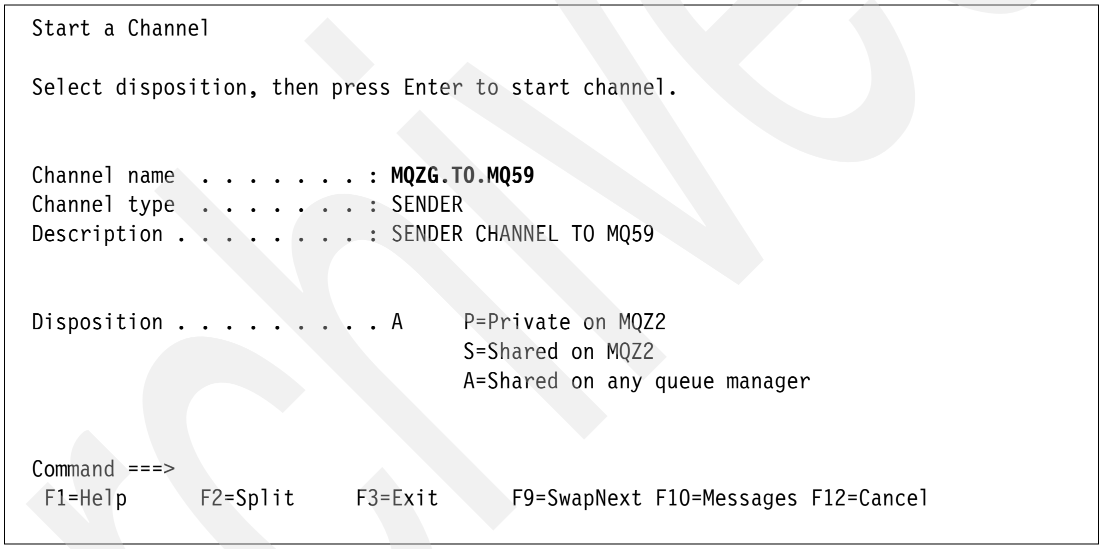
```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
                               Start a Channel

 Select disposition, then press Enter to start channel.


 Channel name  . . . . . . . : MQZG.TO.MQ59
 Channel type  . . . . . . . : RECEIVER
 Description . . . . . . . . : RECEIVER CHANNEL FROM MQZG


 Disposition . . . . . . . . . P     P=Private on MQ59
                                     A=Shared on all queue managers


        EssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssN
        e CSQ9022I -MQ59 CSQXCRPS ' START CHANNEL' NORMAL COMPLETION e
        DssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssM
 Command ===>
  F1=Help      F2=Split     F3=Exit      F9=SwapNext F10=Messages F12=Cancel
```

Figure 4-14   Start the channel between the secondary server SC59 & primary server SC53 3/9



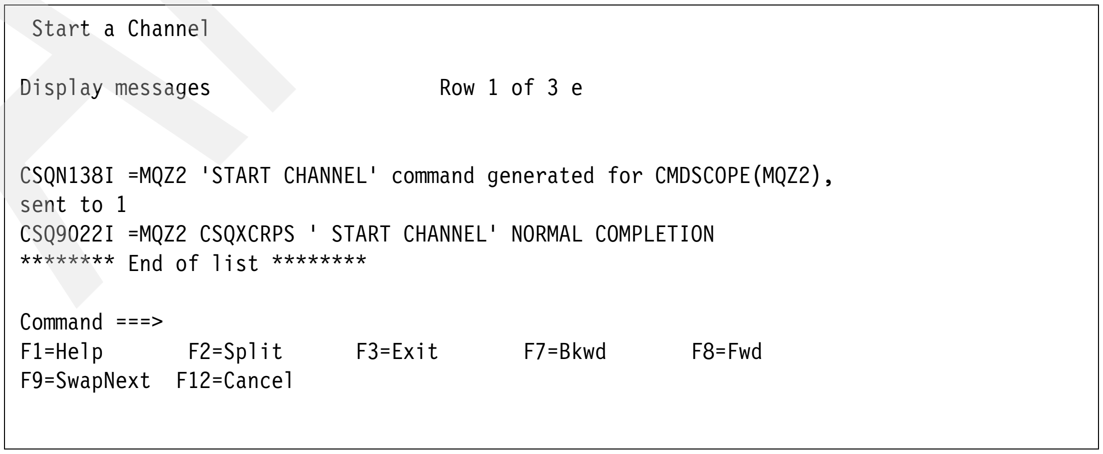
```
List Channels - MQZ2                    Row 1 of 2

Type action codes, then press Enter.  Press F11 to display connection status.
 1=Display   2=Define like   3=Alter   4=Manage   5=Perform
 6=Start     7=Stop

    Name                 Type          Disposition   Status
<>  MQZG.TO.MQ59         CHANNEL       ALL     MQZ2
6   MQZG.TO.MQ59         SENDER        COPY    MQZ2  RETRY
    MQZG.TO.MQ59         SENDER        GROUP
                     ******** End of list ********

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F7=Bkwd
 F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 4-15 Start the channel between the secondary server SC59 & primary server SC53 4/9



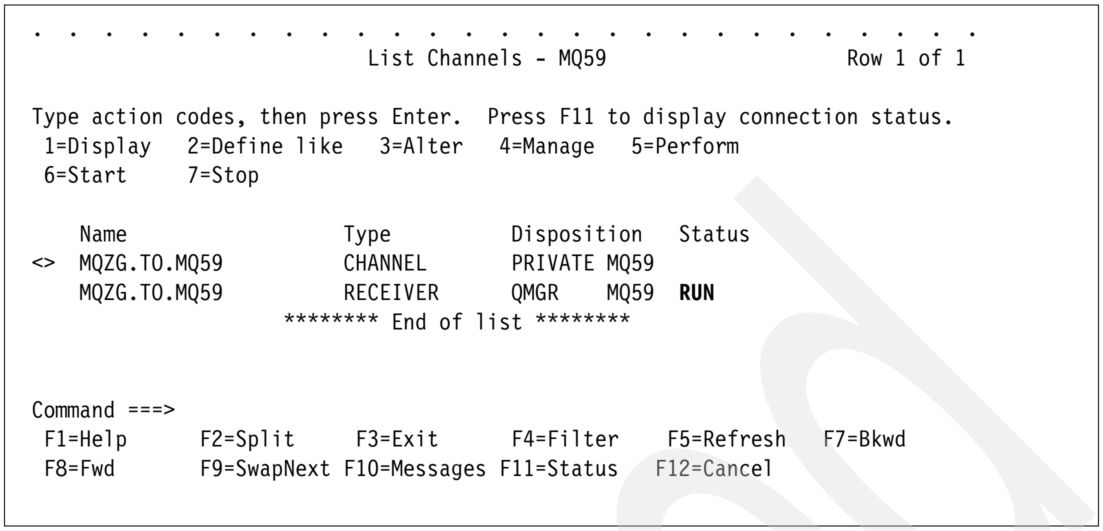
```
Start a Channel

Select disposition, then press Enter to start channel.


Channel name  . . . . . . . : MQZG.TO.MQ59
Channel type  . . . . . . . : SENDER
Description . . . . . . . . : SENDER CHANNEL TO MQ59


Disposition . . . . . . . . . A     P=Private on MQZ2
                                    S=Shared on MQZ2
                                    A=Shared on any queue manager



Command ===>
 F1=Help      F2=Split     F3=Exit      F9=SwapNext F10=Messages F12=Cancel
```

Figure 4-16 Start the channel between the secondary server SC59 & primary server SC53 5/9



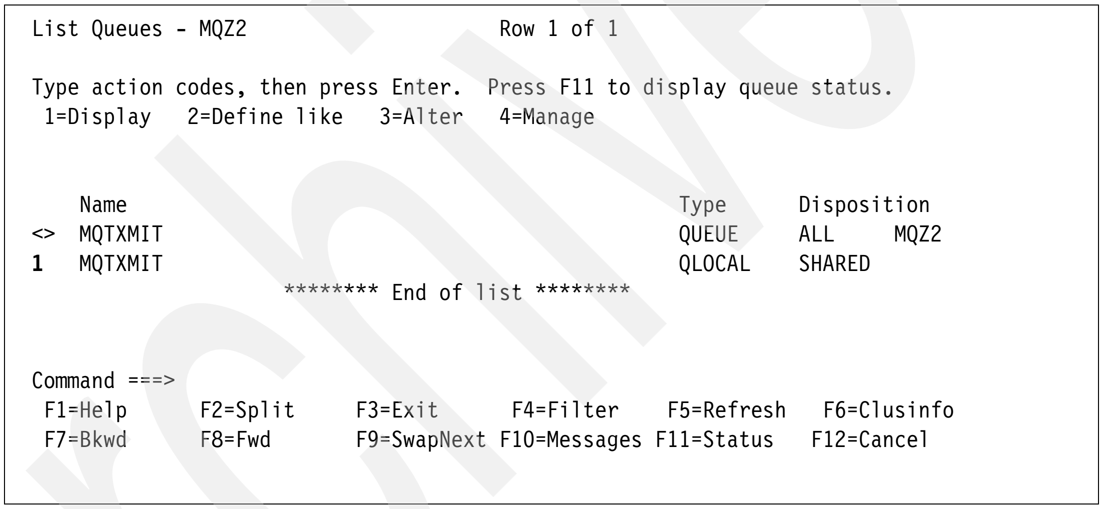
```
 Start a Channel

Display messages                   Row 1 of 3 e



CSQN138I =MQZ2 'START CHANNEL' command generated for CMDSCOPE(MQZ2),
sent to 1
CSQ9022I =MQZ2 CSQXCRPS ' START CHANNEL' NORMAL COMPLETION
******** End of list ********

Command ===>
F1=Help       F2=Split      F3=Exit       F7=Bkwd       F8=Fwd
F9=SwapNext  F12=Cancel
```

Figure 4-17 Start the channel between the secondary server SC59 & primary server SC53 6/9



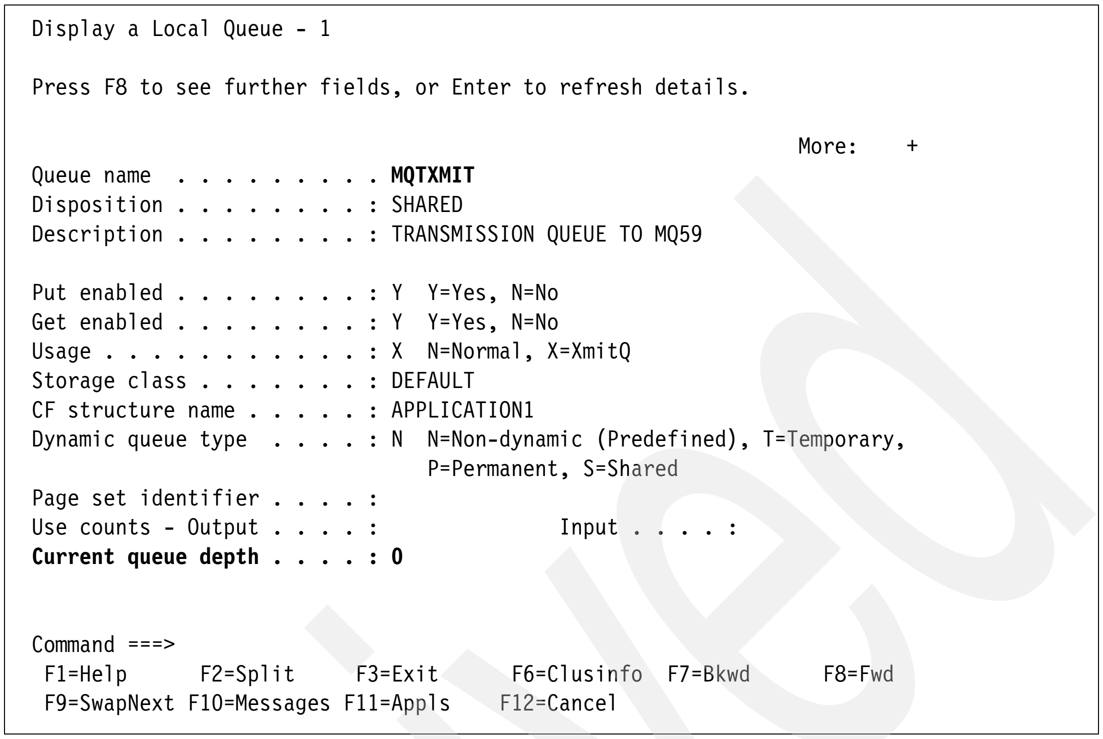
```
 . . . . . . . . . . . . . . . . . . . . . . . . . . .
                            List Channels - MQ59                   Row 1 of 1

 Type action codes, then press Enter.  Press F11 to display connection status.
  1=Display   2=Define like   3=Alter   4=Manage   5=Perform
  6=Start     7=Stop

     Name                  Type          Disposition   Status
 <>  MQZG.TO.MQ59          CHANNEL       PRIVATE MQ59
     MQZG.TO.MQ59          RECEIVER      QMGR    MQ59  RUN
                     ******** End of list ********


 Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F7=Bkwd
  F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 4-18   Start the channel between the secondary server SC59 & primary server SC53 7/9



```
 List Queues - MQZ2                    Row 1 of 1

 Type action codes, then press Enter.  Press F11 to display queue status.
  1=Display   2=Define like   3=Alter   4=Manage


     Name                                          Type      Disposition
 <>  MQTXMIT                                       QUEUE     ALL     MQZ2
 1   MQTXMIT                                       QLOCAL    SHARED
                     ******** End of list ********


 Command ===>
  F1=Help      F2=Split     F3=Exit      F4=Filter    F5=Refresh   F6=Clusinfo
  F7=Bkwd      F8=Fwd       F9=SwapNext F10=Messages F11=Status   F12=Cancel
```

Figure 4-19   Start the channel between the secondary server SC59 & primary server SC53 8/9



```
Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

                                                              More:    +
Queue name  . . . . . . . . . MQTXMIT
Disposition . . . . . . . . : SHARED
Description . . . . . . . . : TRANSMISSION QUEUE TO MQ59

Put enabled . . . . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . . . . : Y  Y=Yes, N=No
Usage . . . . . . . . . . . : X  N=Normal, X=XmitQ
Storage class . . . . . . . : DEFAULT
CF structure name . . . . . : APPLICATION1
Dynamic queue type  . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                 P=Permanent, S=Shared
Page set identifier . . . . :
Use counts - Output . . . . :              Input . . . . :
Current queue depth . . . . : 0


Command ===>
 F1=Help      F2=Split     F3=Exit      F6=Clusinfo  F7=Bkwd      F8=Fwd
 F9=SwapNext F10=Messages F11=Appls    F12=Cancel
```

Figure 4-20 Start the channel between the secondary server SC59 & primary server SC53 9/9

STEP TESTSHQA9: Warm start Q Capture on SC67

In this step, Q Capture is started manually on LPAR SC67, similar to that shown in Example 4-21 on page 133, except that the process is started on LPAR SC67. Its successful initialization can be determined by viewing the contents of the Q Capture log, as described in Example 3-16 on page 50.

STEP TESTSHQA10: Verify all workload changes applied to target tables

With the failover completed, verify that all the workload changes that occurred on the primary server databases D8G1 and D8G2 have been propagated to the target tables on the secondary server SC59. This is achieved by viewing the contents of the tables on the primary server and the secondary server, as shown in Example 4-32 and Example 4-33, respectively.

Attention: The contents of the tables on the primary server and secondary server are identical, indicating that replication continued seamlessly without any data loss even on the failure of LPAR SC53. The only procedure required was to warm start Q Capture on the other LPAR SC67 either manually or automatically.

Example 4-32 Contents of tables in database D8G2 on the primary server SC67

```
connect to d8g2 user mwiman using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = MWIMAN
Local database alias   = D8G2

select * from itsoq.products order by product_id

PRODUCT_ID  PRODUCT_NAME         PRODUCT_CATEGORY         PRICE     MANUFACT_ID
----------- -------------------- ---------------- ----------------- -----------
```

```
10          WIDGETS             TOOLS                             1.00         100
20          THINGAMABOBBERS     TOOLS                             8.00         200
30          WHATCHAMACALLITS    TOOLS                            10.00         100
40          DOODADS             TOOLS                             3.00         100
60          HEAVY HAMMER        TOOLS                           100.00         100
80          FLAMETHROWERS       WEAPONS                        1000.00         300
90          STRAWMEN            TOOLS                             8.00         100
7 record(s) selected.

select * from itsoq.sales order by region_code, product_id, store_id, date_of_sale

PRODUCT_ID  STORE_ID    QUANTITY_SOLD DATE_OF_SALE REGION_CODE
----------- ----------- ------------- ------------ -----------
10         100         1000          2006-03-01   EAST
20         120         2000          2006-03-01   EAST
30         100         5000          2006-03-02   EAST
60         160         6000          2006-03-03   EAST
10         200            0          2006-03-02   WEST
20         220         2000          2006-03-02   WEST
30         200         5000          2006-03-01   WEST
30         200         3000          2006-03-03   WEST
40         140         4000          2006-03-02   WEST
40         240         6000          2006-03-01   WEST
80         270            5          2006-03-04   WEST
11 record(s) selected.

terminate

DB20000I  The TERMINATE command completed successfully.
```

Example 4-33 Contents of tables in database DB8A on the secondary server SC59

```
connect to db8a user mwiman using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = MWIMAN
Local database alias   = DB8A

select * from itsoq.products order by product_id

PRODUCT_ID  PRODUCT_NAME         PRODUCT_CATEGORY           PRICE   MANUFACT_ID
----------- -------------------- ---------------- ----------------- -----------
10          WIDGETS             TOOLS                             1.00         100
20          THINGAMABOBBERS     TOOLS                             8.00         200
30          WHATCHAMACALLITS    TOOLS                            10.00         100
40          DOODADS             TOOLS                             3.00         100
60          HEAVY HAMMER        TOOLS                           100.00         100
80          FLAMETHROWERS       WEAPONS                        1000.00         300
90          STRAWMEN            TOOLS                             8.00         100
7 record(s) selected.

select * from itsoq.sales_east order by product_id, store_id, date_of_sale

PRODUCT_ID  STORE_ID    QUANTITY_SOLD DATE_OF_SALE
----------- ----------- ------------- ------------
10         100         1000          2006-03-01
20         120         2000          2006-03-01
30         100         5000          2006-03-02
60         160         6000          2006-03-03
4 record(s) selected.
```

```
select * from itsoq.sales_west order by product_id, store_id, date_of_sale

PRODUCT_ID  STORE_ID    QUANTITY_SOLD DATE_OF_SALE
----------- ----------- ------------- ------------
10          200            0          2006-03-02
20          220         2000          2006-03-02
30          200         5000          2006-03-01
30          200         3000          2006-03-03
40          140         4000          2006-03-02
40          240         6000          2006-03-01
80          270            5          2006-03-04
7 record(s) selected.

terminate

DB20000I  The TERMINATE command completed successfully.
```

4.6.2 Failover operations with inflight automatic load in progress

The objective of this test case is to demonstrate that under failover operations with inflight automatic load in progress, Q replication operates in a seamless fashion in our WebSphere MQ shared queues high availability primary server environment with no data loss. Some additional steps need to be added to cope with inflight automatic loads that failed when the LPAR it was running on crashed.

Attention: If the automatic load happens to be running on a member of a DB2 data sharing group that is different from the one that has Q Capture running, then the crash of the LPAR running Q Capture does *not* impact the normal completion of the automatic load.

The following assumptions are made about the test case environment:

- Application workloads are running against both members of the DB2 data sharing group D8GG comprising member databases D8G1 on LPAR SC53 and D8G2 on LPAR SC67.
- Q Capture is running on the primary server LPAR SC53.
- Q Apply is running on the secondary server LPAR SC59.
- The PRODUCTS table is replicated in its entirety to the secondary server.
- The source SALES table is replicated to a SALES_EAST and a SALES_WEST table on the target database depending on the REGION_CODE value on the source table. The REGION_CODE column is not replicated.

Figure 4-21 shows the steps executed in this test case.

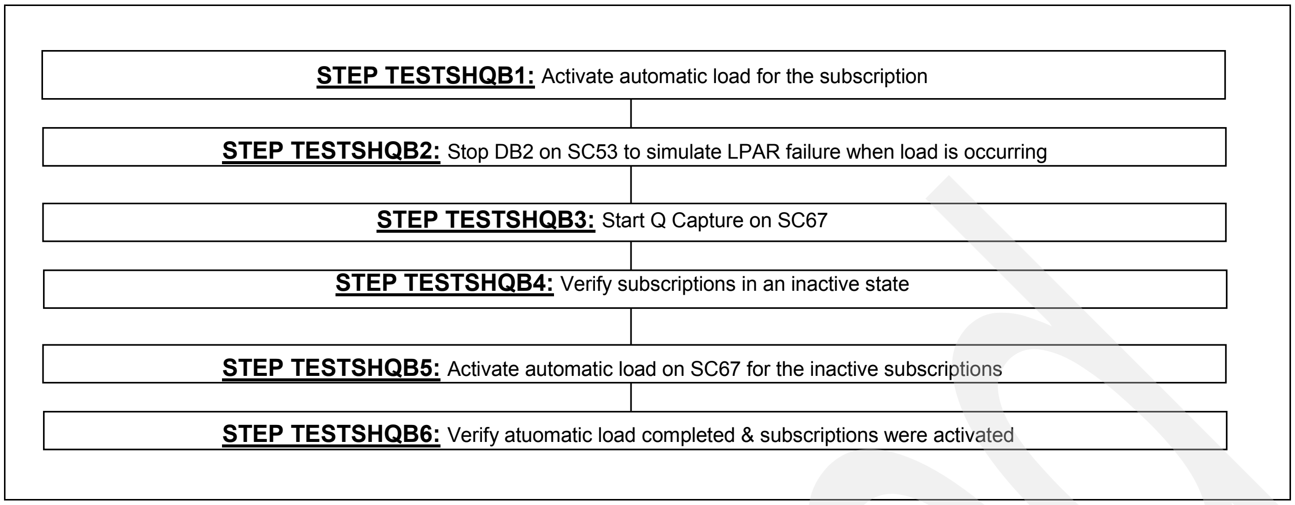

STEP TESTSHQB1: Activate automatic load for the subscription
STEP TESTSHQB2: Stop DB2 on SC53 to simulate LPAR failure when load is occurring
STEP TESTSHQB3: Start Q Capture on SC67
STEP TESTSHQB4: Verify subscriptions in an inactive state
STEP TESTSHQB5: Activate automatic load on SC67 for the inactive subscriptions
STEP TESTSHQB6: Verify atuomatic load completed & subscriptions were activated

Figure 4-21 Failover operations with inflight automatic load in progress test case with shared queues

Each of the steps shown in Figure 4-21 is described in more detail here.

Attention: We have not included the steps that verify that the content on the primary server and secondary server match, since this was demonstrated in Section 4.6.1, “Failover operations with no inflight automatic load in progress” on page 135.

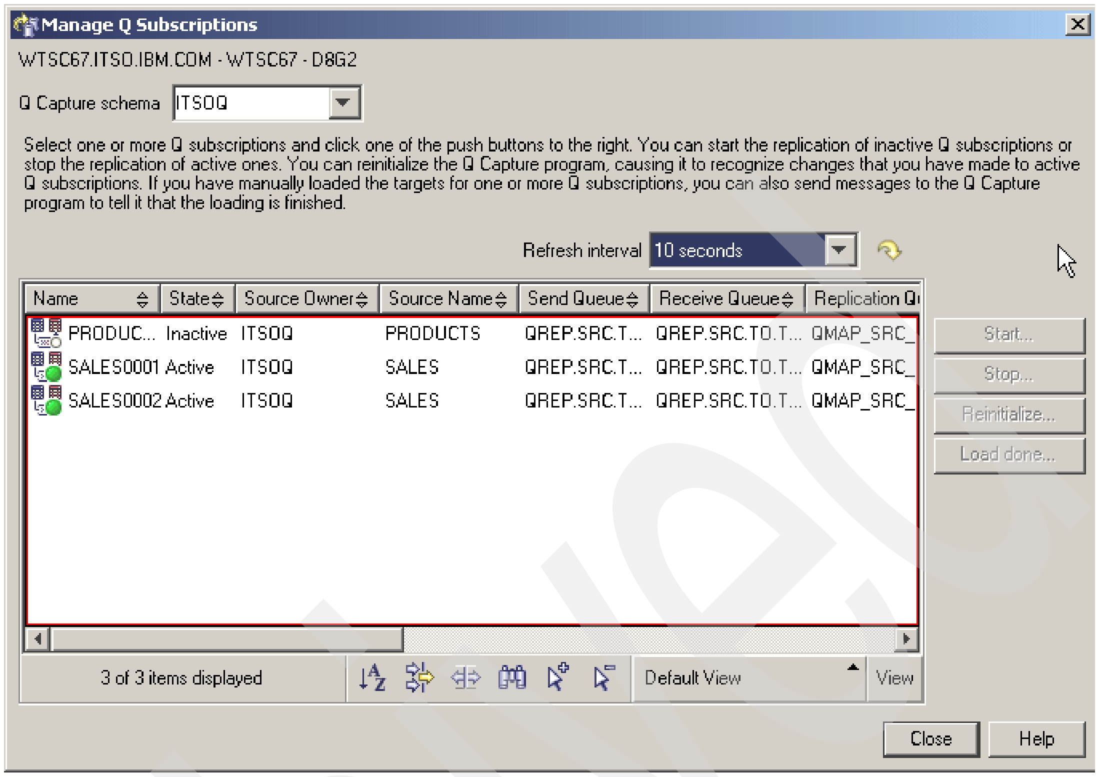
STEP TESTSHQB1: Activate automatic load for the subscription

In this step, activate the PRODUCTS0001 subscription that is inactive, as shown in Figure 4-22, and has been defined to be loaded automatically. Over a million rows were loaded into the PRODUCTS table in the primary server SC53 in order to be able to crash LPAR SC53 while the automatic load of the corresponding table on the secondary server SC59 was in progress.

Note: We took certain steps (not documented here) to ensure that the load was running on the same LPAR SC53 where Q Capture was running.

The PRODUCTS0001 subscription was activated by inserting a CAPSTART signal for it in the IBMQREP_SIGNAL table in the D8G1 database on the primary server SC53, as shown in Example 4-34.

Figure 4-23 shows the PRODUCTS0001 subscription being in the Loading state when automatic load has begun.


Manage Q Subscriptions

WTSC67.ITSO.IBM.COM - WTSC67 - D8G2

Q Capture schema: ITSOQ

Select one or more Q subscriptions and click one of the push buttons to the right. You can start the replication of inactive Q subscriptions or stop the replication of active ones. You can reinitialize the Q Capture program, causing it to recognize changes that you have made to active Q subscriptions. If you have manually loaded the targets for one or more Q subscriptions, you can also send messages to the Q Capture program to tell it that the loading is finished.

Refresh interval: 10 seconds

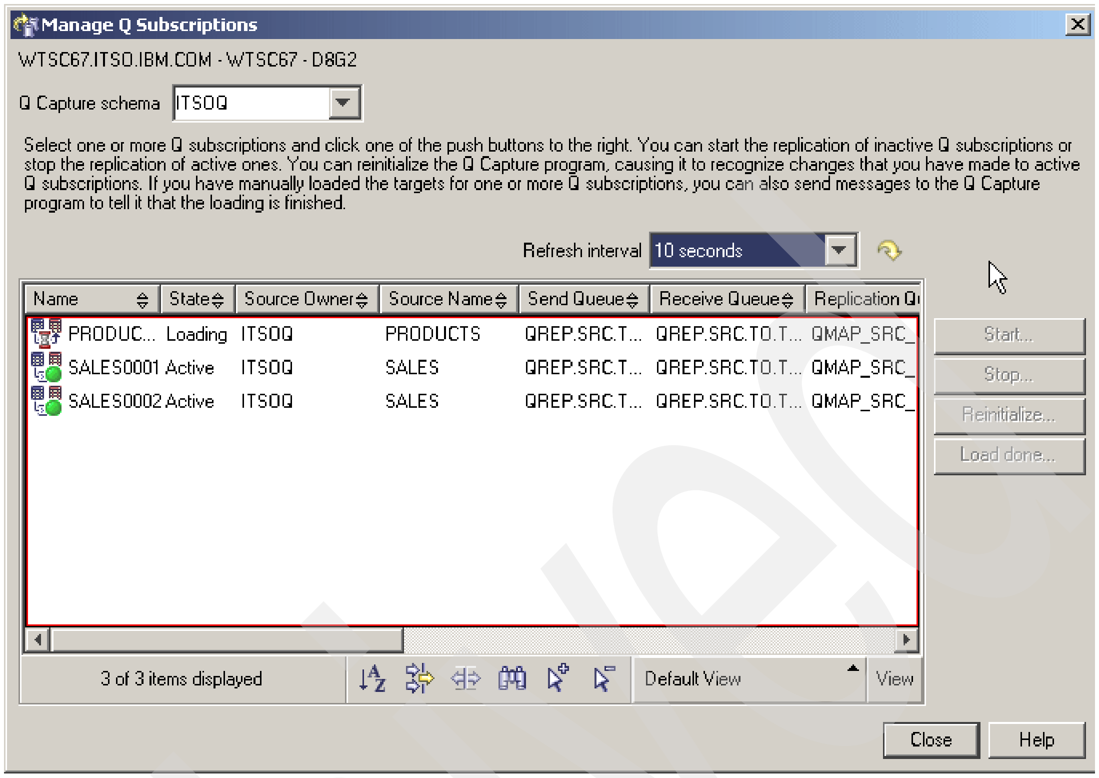
Name	State	Source Owner	Source Name	Send Queue	Receive Queue	Replication Q
PRODUC...	Inactive	ITSOQ	PRODUCTS	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0001	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0002	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_

3 of 3 items displayed

Figure 4-22 Subscription PRODUCTS0001 is inactive using Replication Center

Example 4-34 Activate the subscription PRODUCTS0001 on the D8G1 database on primary server SC53

```
connect to d8g1 user hueykim using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = HUEYKIM
Local database alias   = D8G1

insert into itsoq.ibmqrep_signal (  signal_time,signal_type, signal_subtype,
signal_input_in,   signal_state ) values ( current timestamp,   'CMD',
'CAPSTART','PRODUCTS0001','P')

DB20000I  The SQL command completed successfully.

connect reset

DB20000I  The SQL command completed successfully.
```


Manage Q Subscriptions

WTSC67.ITSO.IBM.COM - WTSC67 - D8G2

Q Capture schema: ITSOQ

Select one or more Q subscriptions and click one of the push buttons to the right. You can start the replication of inactive Q subscriptions or stop the replication of active ones. You can reinitialize the Q Capture program, causing it to recognize changes that you have made to active Q subscriptions. If you have manually loaded the targets for one or more Q subscriptions, you can also send messages to the Q Capture program to tell it that the loading is finished.

Refresh interval: 10 seconds

Name	State	Source Owner	Source Name	Send Queue	Receive Queue	Replication Q
PRODUC...	Loading	ITSOQ	PRODUCTS	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0001	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0002	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_

3 of 3 items displayed

Figure 4-23 Subscription PRODUCTS0001 is loading using Replication Center

STEP TESTSHQB2: Stop DB2 on SC53 to simulate LPAR failure

In this step, stop DB2 on LPAR SC53 to simulate its failure, as shown in Example 4-30 on page 145.

The contents of the Q Capture log on the primary server SC53 (shown in Example 4-35) indicate that the Q Capture program fails after receiving a connection failure return code since DB2 is down.

The contents of the Q Apply log on the secondary server SC59 (shown in Example 4-36 on page 156) indicate that the subscription is deactivated because of the error action option chosen when the load fails as a result of DB2 being unavailable at the primary server SC53.

Example 4-37 on page 157 shows the contents of the load trace file about the failure of the load due to a connection failure.

Example 4-38 on page 158 displays the status of the tablespace in which the PRODUCTS table resides on the secondary server as being in a RECP[1] status after the automatic load failed.

Attention: Whenever an automatic load fails, the target tablespace will be placed in a RECP state. One way to remove this state is to initiate the automatic load again.

Figure 4-24 on page 159 shows the state of the PRODUCTS0001 subscription as remaining in the Loading state.

[1] Recovery pending state

Example 4-35 Contents of Q Capture log on the primary server SC53

```
...........................
2006-03-04-13.19.31.861201 <dbCtxEnd> ASN0552E  "Q Capture" : "ITSOQ" : "HoldLThread" : The
program encountered an SQL error. The server name is "DB8G". The SQL request is "RELEASE
CURRENT". The table name is "N/A". The SQLCODE is "-924". The SQLSTATE is "58006". The
SQLERRMC is "0001Ÿ0100Ÿ00F30018". The SQLERRP is "DSNAET03".
2006-03-04-13.19.31.861201 <dbCtxEnd> ASN8999D  "Q Capture" : "ITSOQ" : "HoldLThread" :
  DSNT408I SQLCODE = -924, ERROR:  DB2 CONNECTION INTERNAL ERROR, 0001,
           0100, 00F30018
  DSNT418I SQLSTATE   = 58006 SQLSTATE RETURN CODE
  DSNT415I SQLERRP    = DSNAET03 SQL PROCEDURE DETECTING ERROR
2006-03-04-13.19.31.877384 <dbConnection::dbCtxEnd> ASN0535E  "Q Capture" : "ITSOQ" :
"HoldLThread" : The program could not disconnect from the database "DB8G". The return code
is "c8", and the reason code is "c12204".
2006-03-04-13.19.31.880247 <HoldLockMain> ASN0589I  "Q Capture" : "ITSOQ" : "HoldLThread"
The program received an unexpected return code "913" from routine "dbConnection::dbCtxEnd".
2006-03-04-13.19.32.863262 <asnqcap::main> ASN0573I  "Q Capture" : "ITSOQ" : "Initial" :
The program was stopped.
2006-03-04-13.19.34.262282 <dbCtxEnd> ASN0552E  "Q Capture" : "ITSOQ" : "Initial" : The
program encountered an SQL error. The server name is "DB8G". The SQL request is "RELEASE
CURRENT". The table name is "N/A". The SQLCODE is "-924". The SQLSTATE is "58006". The
SQLERRMC is "0001Ÿ0100Ÿ00F30018". The SQLERRP is "DSNAET03".
2006-03-04-13.19.34.262282 <dbCtxEnd> ASN8999D  "Q Capture" : "ITSOQ" : "Initial" :
  DSNT408I SQLCODE = -924, ERROR:  DB2 CONNECTION INTERNAL ERROR, 0001,
           0100, 00F30018
  DSNT418I SQLSTATE   = 58006 SQLSTATE RETURN CODE
  DSNT415I SQLERRP    = DSNAET03 SQL PROCEDURE DETECTING ERROR
2006-03-04-13.19.34.544192 <dbConnection::dbCtxEnd> ASN0535E  "Q Capture" : "ITSOQ" :
"Initial" : The program could not disconnect from the database "DB8G". The return code is
"c8", and the reason code is "c12204".
2006-03-04-13.19.34.591829 <asnqcap::main> ASN0589I  "Q Capture" : "ITSOQ" : "Initial" The
program received an unexpected return code "913" from routine "dbConnection::dbCtxEnd".
```

Example 4-36 Contents of Q Apply log on the secondary server SC59

```
.................
The Q Apply program for the Q subscription "PRODUCTS0001" (receive queue
"QREP.SRC.TO.TRG.RECVQ", replication queue map "QMAP_SRC_TO_TRG") will use the "LOAD from
CURSOR" utility to load table "ITSOQ.PRODUCTS".
2006-03-04-13.19.26.388676 <invokeLoad> ASN7530E  "Q Apply" : "ITSOQ" : "BR00000SP003" :
The load utility "LOAD from CURSOR" for table "ITSOQ.PRODUCTS" failed for Q subscription
"PRODUCTS0001" (receive queue "QREP.SRC.TO.TRG.RECVQ", replication queue map
"QMAP_SRC_TO_TRG"). Detailed message from the load utility is "RC: 8,
//'HUEYKIM.DB8A.ITSOQ.S0000003.LOADTRC'".
2006-03-04-13.19.26.397815 <spillAgntMain> ASN0589I  "Q Apply" : "ITSOQ" : "BR00000SP003"
The program received an unexpected return code "8" from routine "QAsub::callLoader".
2006-03-04-13.19.26.401867 <spillAgntMain> ASN7597E  "Q Apply" : "ITSOQ" : "BR00000SP003" :
The Q subscription "PRODUCTS0001" (receive queue "QREP.SRC.TO.TRG.RECVQ", replication queue
map "QMAP_SRC_TO_TRG") is about to be disabled because of the conflict action or error
action.
2006-03-04-13.19.26.401867 <spillAgntMain> ASN8999D  "Q Apply" : "ITSOQ" : "BR00000SP003" :
disable sub according to error/conflict action
2006-03-04-13.19.28.722576 <cleanupSpillQ> ASN8999D  "Q Apply" : "ITSOQ" : "BR00000" :
spillQ 'IBMQREP.SPILL.MODELQ.0.3.1' for sub 'PRODUCTS0001' successfully deleted
2006-03-04-13.19.48.733812 <asnThread::stop> ASN0590I  "Q Apply" : "ITSOQ" : "BR00000" The
thread "BR00000" received return code "2011" from the exiting thread "BR00000SP003".
```

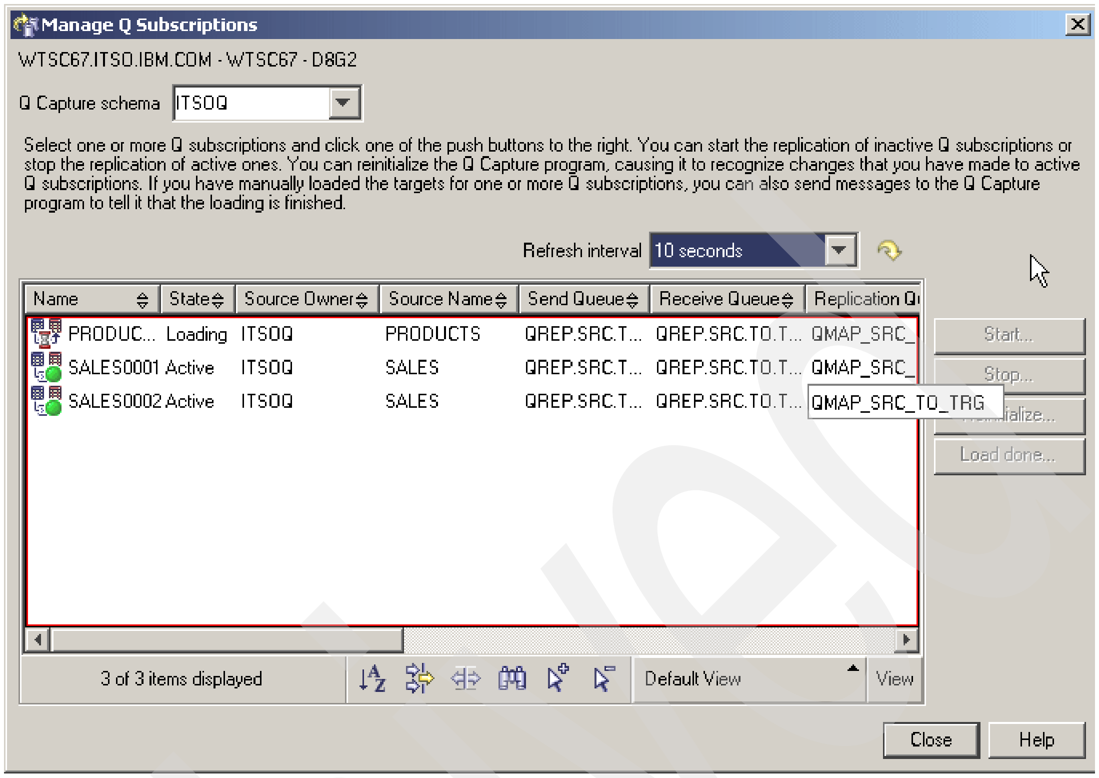
Example 4-37 Contents of the load trace file generated by the automatic load

```
The current timestamp is 2006-03-04-13.18.40.635523

ASNQLOAD(compiled at 11:40:03 on Jan 19 2006) input values:
   Target Server  = DB8A
   Target Owner   = ITSOQ
   Target Table.  = PRODUCTS
   Source Server  = DB8G
   SQL.Stmt       = SELECT "MANUFACT_ID", "PRICE", "PRODUCT_CATEGORY", "PRODUCT_NAME",
"PRODUCT_ID" FROM "ITSOQ"."PRODUCTS"
Connect to TRG DB8A
--> INDEXES  == 1
--> FOR_KEYS == 0
--> getSourceTable() = "ITSOQ"."PRODUCTS"
Connect to SRC DB8G
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "ITSOQ"."PRODUCTS"
--> on server = DB8G
--> RET = 0
--> num_rows   == 1000000
--> getSortKeysTotal() == 1000000

--------> Running LOAD...

ASNLOAD Running under user = MWIMAN
sIdBuf is 00000003
uIdBuf is ITSOQ00000003
WorkDSN1 = MWIMAN.S0000003.UNLDEI
WorkDSN2 = MWIMAN.S0000003.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDE MAXPRIME 4300 TEMPLATE WORKDSN1
DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEI
SPACE (50,100) CYL MAXPRIME 4300 TEMPLATE WORKDSN2 DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEO
SPACE (50,100) CYL MAXPRIME 4300 EXEC S
QL DECLARE C1 CURSOR FOR SELECT "MANUFACT_ID", "PRICE", "PRODUCT_CATEGORY", "PRODUCT_NAME",
"PRODUCT_ID" FROM "DB8G"."ITSOQ"."PROD
UCTS"  ENDEXEC LOAD DATA INCURSOR(C1)  LOG NO NOCOPYPEND  WORKDDN(WORKDSN1,WORKDSN2)
REPLACE SORTKEYS 1000000 INTO TABLE "ITSOQ".
"PRODUCTS" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ITSOQ00000003
 DSNU1044I   DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDE MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEI SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEO SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "MANUFACT_ID", "PRICE",
"PRODUCT_CATEGORY",
 "PRODUCT_NAME", "PRODUCT_ID" FROM "DB8G"."ITSOQ"."PRODUCTS" ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) LOG NO NOCOPYPEND WORKDDN(WORKDSN1,
WORKDSN2) REPLACE SORTKEYS
 1000000
 DSNU650I  -DB8A DSNURWI -  INTO TABLE "ITSOQ"."PRODUCTS" STATISTICS
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN1
                        DDNAME=SYS00005
```

```
                       DSN=HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEI
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN2
                       DDNAME=SYS00006
                       DSN=HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEO
 DSNU350I  -DB8A DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU1188I   DSNUGSQL -  SQLCODE = -30081, ERROR:   COMMUNICATIONS ERROR DETECTED.  API=,
LOCATION=, FUNCTION=
                        ERROR CODES=
 DSNU1198I   DSNUGSQL -  SQLSTATE   = 08001 SQLSTATE RETURN CODE
 DSNU1195I   DSNUGSQL -  SQLERRP    = DSNLCSRR SQL PROCEDURE DETECTING ERROR
 DSNU1196I   DSNUGSQL -  SQLERRD    = 9  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
 DSNU1196I   DSNUGSQL -  SQLERRD    = X'00000009'  X'00000000'  X'00000000'  X'FFFFFFFF'
X'00000000'  X'00000000' SQL
                         DIAGNOSTIC INFORMATION
 DSNU304I  -DB8A DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=576400 FOR TABLE
ITSOQ.PRODUCTS
 DSNU1147I -DB8A DSNURWT - (RE)LOAD PHASE STATISTICS - TOTAL NUMBER OF RECORDS
LOADED=576400 FOR TABLESPACE
 DSNDB04.PRODUCTS
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS
PROCESSED=576400
 DSNU562I  -DB8A DSNUGSRX - TABLESPACE DSNDB04.PRODUCTS IS IN RECOVER PENDING
 DSNU830I  -DB8A DSNUGSRX - INDEX ITSOQ.PRODUCT_PK IS IN REBUILD PENDING
 DSNU012I    DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
--------> LOAD ended with CODE = 8

Exiting ASNLOAD
```

Example 4-38   Display status of the PRODUCTS table tablespace on the secondary server SC59

```
-DISPLAY DB(DSNDB04) SPACE(PRODUCTS)

 DSNT360I  -DB8A ***********************************
 DSNT361I  -DB8A *  DISPLAY DATABASE SUMMARY
                 *    GLOBAL
 DSNT360I  -DB8A ***********************************
 DSNT362I  -DB8A     DATABASE = DSNDB04  STATUS = RW
                    DBD LENGTH = 28256
 DSNT397I  -DB8A
 NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
 -------- ---- ----- ----------------- -------- -------- -------- -----
 PRODUCTS TS         RW,RECP
 ******* DISPLAY OF DATABASE DSNDB04  ENDED      **********************
 DSN9022I  -DB8A DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
 ***
```


Manage Q Subscriptions

WTSC67.ITSO.IBM.COM - WTSC67 - D8G2

Q Capture schema: ITSOQ

Select one or more Q subscriptions and click one of the push buttons to the right. You can start the replication of inactive Q subscriptions or stop the replication of active ones. You can reinitialize the Q Capture program, causing it to recognize changes that you have made to active Q subscriptions. If you have manually loaded the targets for one or more Q subscriptions, you can also send messages to the Q Capture program to tell it that the loading is finished.

Refresh interval: 10 seconds

Name	State	Source Owner	Source Name	Send Queue	Receive Queue	Replication Q
PRODUC...	Loading	ITSOQ	PRODUCTS	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0001	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0002	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_TO_TRG

3 of 3 items displayed

Figure 4-24 Subscription PRODUCTS0001 in Loading state using Replication Center

STEP TESTSHQB3: Start Q Capture on primary server SC67

In this step, Q Capture is started manually on LPAR SC67, similar to that shown in Example 4-21 on page 133, except that the process is started on LPAR SC67. Its successful initialization can be determined by viewing the contents of the Q Capture log, as described in Example 3-16 on page 50.

STEP TESTSHQB4: Verify subscriptions in an inactive state

As discovered from the Q Apply log in Example 4-36 on page 156 in “STEP TESTSHQB2: Stop DB2 on SC53 to simulate LPAR failure” on page 155, the subscription is disabled (inactivated) on encountering an error.

Attention: All subscriptions that may have been inactivated by the Q Apply program due to a failure of an inflight automatic load must be identified and activated again.

Example 4-39 shows the state of the subscriptions in the IBMQREP_SUBS table on the D8G2 database on the primary server SC67 indicating that the PRODUCTS0001 subscription is in an inactive state.

Example 4-40 shows the state of the subscription in the IBMQREP_TARGETS table on the DB8A database on the secondary server SC59 table indicating that the PRODUCTS0001 subscription is in an inactive state.

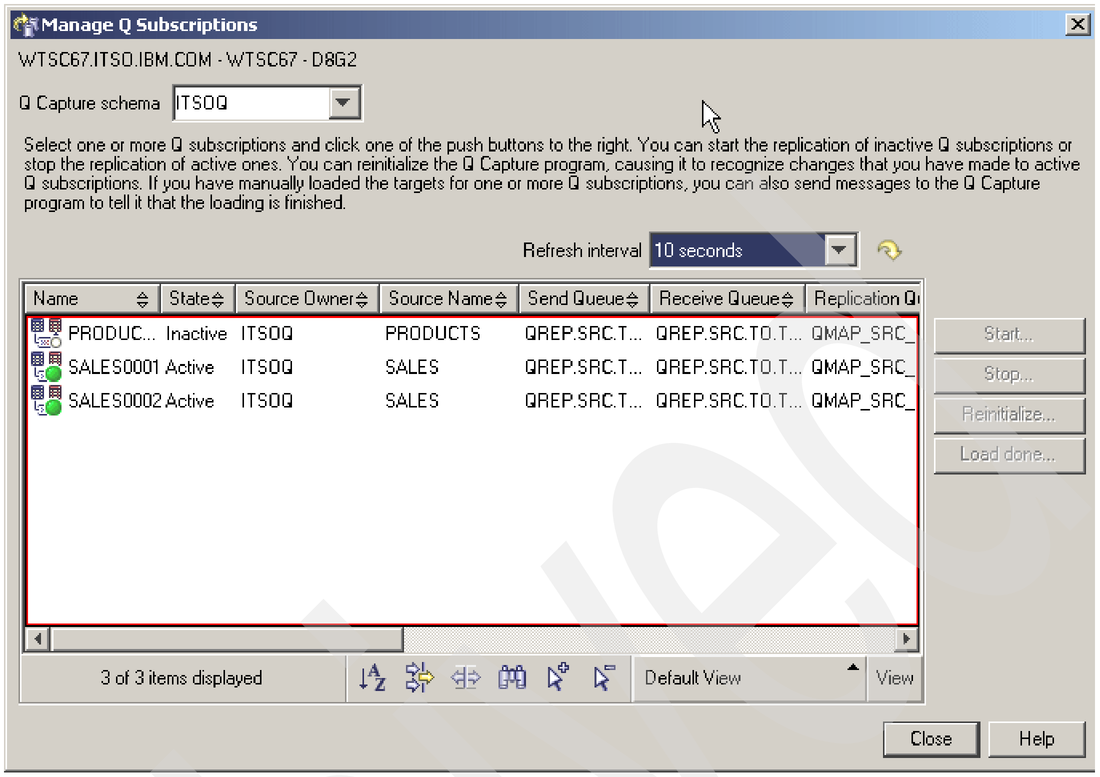
Figure 4-25 on page 161 shows the PRODUCTS001 subscription state as being inactive in the Replication Center.

Example 4-39 Identify inactive subscriptions using the IBMQREP_SUBS table on the primary server SC53

```
connect to d8g2 user mwiman using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = MWIMAN
Local database alias   = D8G2

select
subname,source_owner,source_name,target_server,target_name,sub_id,changed_cols_only,has_loadphase,state
from itsoq.ibmqrep_subs

SUBNAME SOURCE_OWNER SOURCE_NAME TARGET_SERVER TARGET_NAME SUB_ID CHANGED_COLS_ONLY HAS_LOADPHASE STATE
-------------------------------------------------------------------------------------------------------------

PRODUCTS0001 ITSOQ     PRODUCTS    DB8A          PRODUCTS    3         Y                 I          I
SALES0001    ITSOQ     SALES       DB8A          SALES_EAST  2         Y                 I          A
SALES0002    ITSOQ     SALES       DB8A          SALES_WEST  1         Y                 I          A
3 record(s) selected.

terminate

DB20000I  The TERMINATE command completed successfully.
```

Example 4-40 Subscriptions state in IBMQREP_TARGETS table on the secondary server SC59

```
connect to db8a user mwiman using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = MWIMAN
Local database alias   = DB8A

select subname,sub_id,source_server,source_owner,source_name,target_owner,target_name,state from
itsoq.ibmqrep_targets

SUBNAME     SUB_ID SOURCE_SERVER SOURCE_OWNER SOURCE_NAME TARGET_OWNER TARGET_NAME   STATE
-------------------------------------------------------------------------------------------------------------
PRODUCTS0001 3          DB8G          ITSOQ      PRODUCTS      ITSOQ      PRODUCTS      I
SALES0001    2          DB8G          ITSOQ      SALES         ITSOQ      SALES_EAST    A
SALES0002    1          DB8G          ITSOQ      SALES         ITSOQ      SALES_WEST    A
3 record(s) selected.

terminate

DB20000I  The TERMINATE command completed successfully.
```


Manage Q Subscriptions

WTSC67.ITSO.IBM.COM - WTSC67 - D8G2

Q Capture schema: ITSOQ

Select one or more Q subscriptions and click one of the push buttons to the right. You can start the replication of inactive Q subscriptions or stop the replication of active ones. You can reinitialize the Q Capture program, causing it to recognize changes that you have made to active Q subscriptions. If you have manually loaded the targets for one or more Q subscriptions, you can also send messages to the Q Capture program to tell it that the loading is finished.

Refresh interval: 10 seconds

Name	State	Source Owner	Source Name	Send Queue	Receive Queue	Replication Q
PRODUC...	Inactive	ITSOQ	PRODUCTS	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0001	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0002	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_

3 of 3 items displayed

Figure 4-25 Subscriptions state using Replication Center

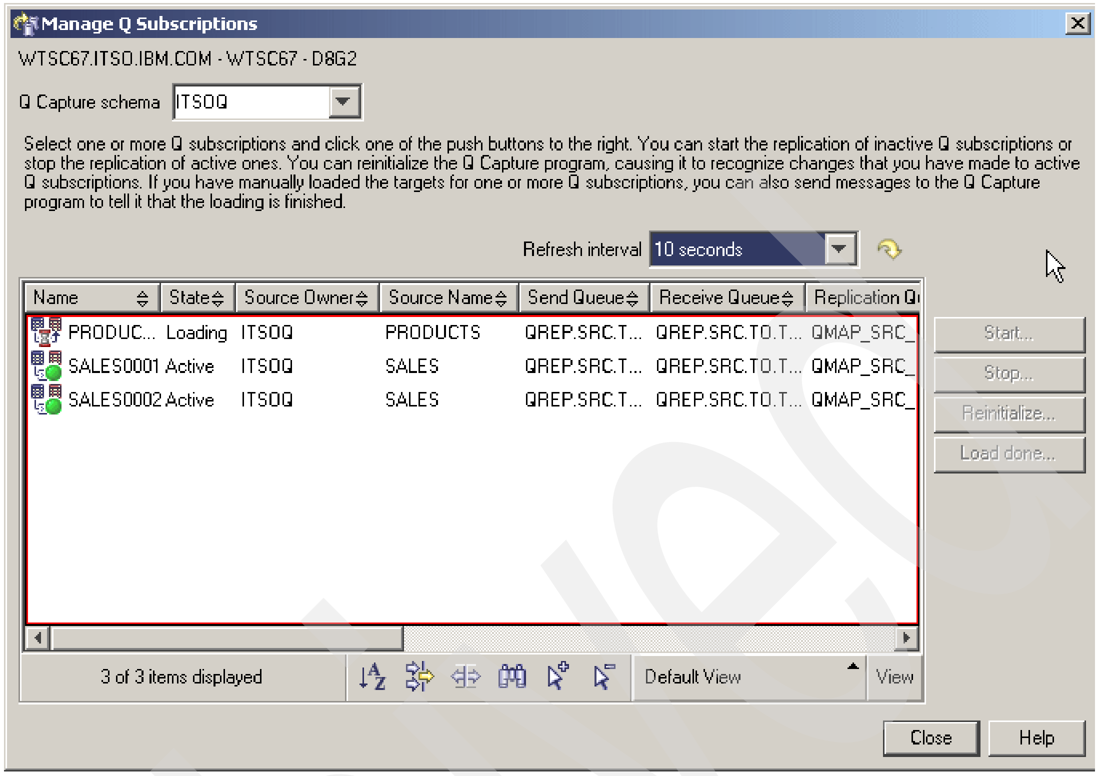
STEP TESTSHQB5: Activate automatic load on SC67 for inactive subs

After identifying the subscriptions that were inactivated by the Q Apply program, you must activate them by issuing a CAPSTART signal in the IBMQREP_SIGNAL table in the D8G1 database on the primary server SC67, as shown in Example 4-41.

After the CAPSTART signal, the subscription initiates the automatic load. Figure 4-26 shows the PRODUCTS0001 subscription being in the loading state using the Replication Center.

Example 4-41 Activate the PRODUCTS0001 subscription on the D8G2 database on primary server SC67

```
connect to d8g2 user hueykim using
Database Connection Information
Database server        = DB2 OS/390 8.1.5
SQL authorization ID   = HUEYKIM
Local database alias   = D8G2

insert into itsoq.ibmqrep_signal (  signal_time,signal_type, signal_subtype,
signal_input_in,   signal_state ) values ( current timestamp,   'CMD',
'CAPSTART','PRODUCTS0001','P')

DB20000I  The SQL command completed successfully.

connect reset

DB20000I  The SQL command completed successfully.
```


Manage Q Subscriptions

WTSC67.ITSO.IBM.COM - WTSC67 - D8G2

Q Capture schema: ITSOQ

Select one or more Q subscriptions and click one of the push buttons to the right. You can start the replication of inactive Q subscriptions or stop the replication of active ones. You can reinitialize the Q Capture program, causing it to recognize changes that you have made to active Q subscriptions. If you have manually loaded the targets for one or more Q subscriptions, you can also send messages to the Q Capture program to tell it that the loading is finished.

Refresh interval: 10 seconds

Name	State	Source Owner	Source Name	Send Queue	Receive Queue	Replication Q
PRODUC...	Loading	ITSOQ	PRODUCTS	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0001	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_
SALES0002	Active	ITSOQ	SALES	QREP.SRC.T...	QREP.SRC.TO.T...	QMAP_SRC_

3 of 3 items displayed

Figure 4-26 Subscription PRODUCTS0001 in Loading state using Replication Center

STEP TESTSHQB6: Verify automatic load completed and subs activated

In this step, verify the successful completion of the automatic load by viewing the contents of the load trace file, as shown in Example 4-42. The active state of this subscription may be verified using the Replication Center (not shown here).

Example 4-42 Contents of the load trace file

```
The current timestamp is 2006-03-04-14.05.02.240645

ASNQLOAD(compiled at 11:40:03 on Jan 19 2006) input values:
   Target Server  = DB8A
   Target Owner   = ITSOQ
   Target Table.  = PRODUCTS
   Source Server  = DB8G
   SQL.Stmt       = SELECT "MANUFACT_ID", "PRICE", "PRODUCT_CATEGORY", "PRODUCT_NAME",
"PRODUCT_ID" FROM "ITSOQ"."PRODUCTS"
Connect to TRG DB8A
--> INDEXES  == 1
--> FOR_KEYS == 0
--> getSourceTable() = "ITSOQ"."PRODUCTS"
Connect to SRC DB8G
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "ITSOQ"."PRODUCTS"
--> on server = DB8G
--> RET = 0
--> num_rows    == 1000000
--> getSortKeysTotal() == 1000000

--------> Running LOAD...
```

```
ASNLOAD Running under user = MWIMAN
sIdBuf is 00000003
uIdBuf is ITSOQ00000003
WorkDSN1 = MWIMAN.S0000003.UNLDEI
WorkDSN2 = MWIMAN.S0000003.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDE MAXPRIME 4300 TEMPLATE WORKDSN1
DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEI
SPACE (50,100) CYL MAXPRIME 4300 TEMPLATE WORKDSN2 DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEO
SPACE (50,100) CYL MAXPRIME 4300 EXEC S
QL DECLARE C1 CURSOR FOR SELECT "MANUFACT_ID", "PRICE", "PRODUCT_CATEGORY", "PRODUCT_NAME",
"PRODUCT_ID" FROM "DB8G"."ITSOQ"."PROD
UCTS"  ENDEXEC LOAD DATA INCURSOR(C1)  LOG NO NOCOPYPEND  WORKDDN(WORKDSN1,WORKDSN2)
REPLACE SORTKEYS 1000000 INTO TABLE "ITSOQ".
"PRODUCTS" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ITSOQ00000003
 DSNU1044I   DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDE MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEI SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEO SPACE(50,
100) CYL MAXPRIME 4300
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "MANUFACT_ID", "PRICE",
"PRODUCT_CATEGORY",
 "PRODUCT_NAME", "PRODUCT_ID" FROM "DB8G"."ITSOQ"."PRODUCTS" ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) LOG NO NOCOPYPEND WORKDDN(WORKDSN1,
WORKDSN2) REPLACE SORTKEYS
 1000000
 DSNU650I  -DB8A DSNURWI -  INTO TABLE "ITSOQ"."PRODUCTS" STATISTICS
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN1
                         DDNAME=SYS00007
                         DSN=HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEI
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=WORKDSN2
                         DDNAME=SYS00008
                         DSN=HUEYKIM.DB8A.ITSOQ.S0000003.UNLDEO
 DSNU350I  -DB8A DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU304I  -DB8A DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=1000000 FOR TABLE
ITSOQ.PRODUCTS
 DSNU1147I -DB8A DSNURWT - (RE)LOAD PHASE STATISTICS - TOTAL NUMBER OF RECORDS
LOADED=1000000 FOR TABLESPACE
 DSNDB04.PRODUCTS
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS
PROCESSED=1000000
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:01:09
 DSNU042I    DSNUGSOR - SORT PHASE STATISTICS -
                       NUMBER OF RECORDS=1000000
                       ELAPSED TIME=00:00:00
 DSNU349I  -DB8A DSNURBXA - BUILD PHASE STATISTICS - NUMBER OF KEYS=1000000 FOR INDEX
ITSOQ.PRODUCT_PK
 DSNU258I    DSNURBXD - BUILD PHASE STATISTICS - NUMBER OF INDEXES=1
 DSNU259I    DSNURBXD - BUILD PHASE COMPLETE, ELAPSED TIME=00:00:04
 DSNU610I  -DB8A DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR DSNDB04.PRODUCTS SUCCESSFUL
 DSNU610I  -DB8A DSNUSUTB - SYSTABLES CATALOG UPDATE FOR ITSOQ.PRODUCTS SUCCESSFUL
 DSNU610I  -DB8A DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DSNDB04.PRODUCTS SUCCESSFUL
 DSNU620I  -DB8A DSNUSEF2 - RUNSTATS CATALOG TIMESTAMP = 2006-03-04-14.05.05.203336
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
--------> LOAD ended with CODE = 0

Exiting ASNLOAD
```

5

WebSphere MQ shared disk and unidirectional replication

In this chapter we describe a step-by-step approach to implementing a WebSphere MQ shared disk high availability environment on the source system of a unidirectional Q replication environment.

The topics covered include:

- Business requirement
- Rationale for the unidirectional solution
- Environment configuration
- Set up of the environment
- Test cases

5.1 Introduction

As described in Section 4.1, “Introduction” on page 104, unidirectional replication using Q replication may be the replication of choice for environments that require a high-volume, low-latency solution between one or more tables on two servers, with a capability to perform transformations on the target tables. Such scenarios are typical of:

- Data warehousing applications that require simple to complex data transformations of operational data.
- Offloading of query reporting away from critical operational systems, by creating complete or partial subsets of operational data with or without transformations.

In this chapter we describe an existing DB2 for z/OS data-sharing high availability business solution implementation into which is seamlessly introduced a unidirectional Q replication application. This unidirectional Q replication application leverages the shared disk capability of the DB2 for z/OS data sharing environment via WebSphere MQ’s shared disk functionality. A brief overview of the business requirement is followed by a description of the environment configuration. The setup of this WebSphere MQ shared disk HA environment with unidirectional replication is described, followed by test cases showing the successful operation of the application in this environment. The topics covered in the following sections are:

- Business requirement
- Rationale for the unidirectional solution
- Environment configuration
- Set up of this environment
- Test cases

5.2 Business requirement

Our fictitious company AvantGarde is a fashion clothes retail organization with stores spread across the entire continental United States. It also provides its patrons with round-the-clock online shopping facilities. The requirement to provide high availability was met with a DB2 for z/OS data sharing implementation. The increase in the query workload by buyers in the various stores against the existing sales reporting application prompted a need to offload as much of the workload as possible off the critical operational systems.

These requirements may be summarized as follows:

1. Offload query workload against the existing sales reporting application to secondary servers.
2. Devise a solution that coexists synergistically with the existing operational systems’ DB2 for z/OS data sharing implementation without using the scarce resources of the coupling facility used by the DB2 for z/OS data sharing.

5.3 Rationale for the unidirectional solution

IBM’s WebSphere Replication Server for z/OS provides the functionality of high-volume low-latency unidirectional replication for DB2 for z/OS data sources in a data sharing environment. The WebSphere Replication Server for z/OS uses WebSphere MQ as the underlying transport mechanism, and its shared disk functionality is particularly synergistic with the existing HA environment that uses shared disk in its DB2 for z/OS data sharing implementation.

5.4 Environment configuration

Figure 5-1 shows the configuration used in the AvantGarde unidirectional replication topology.

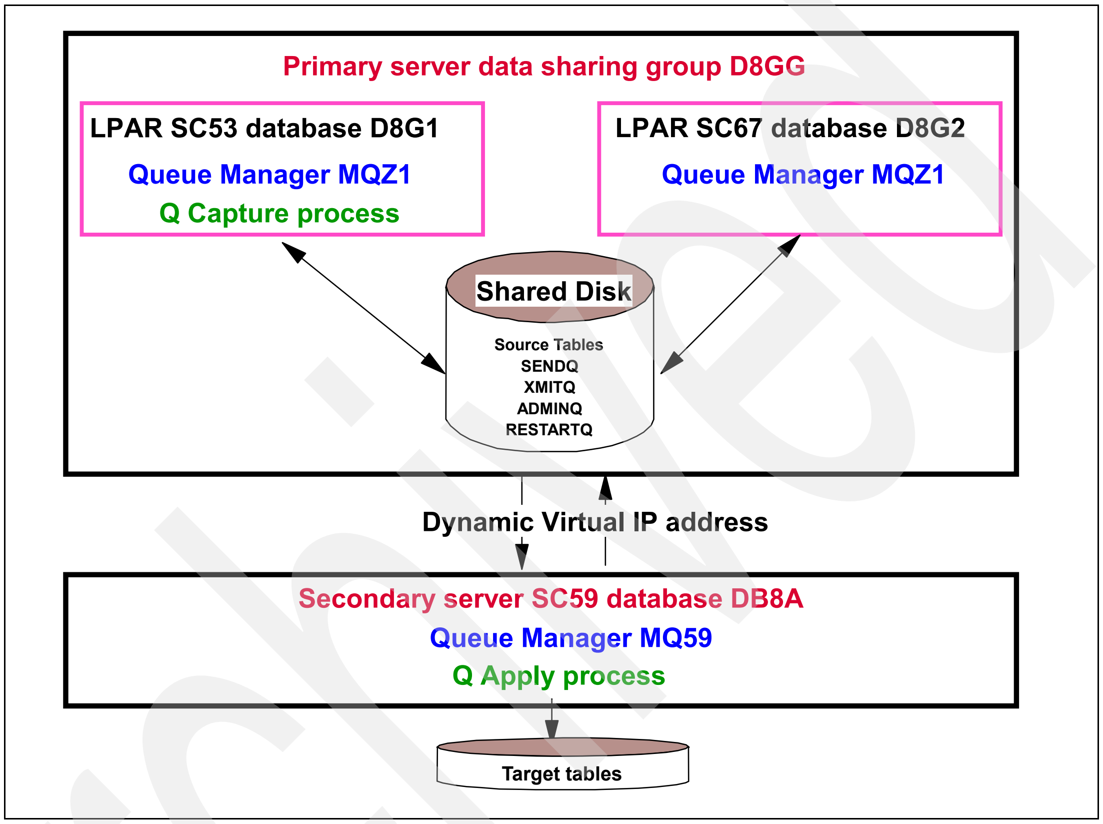


Figure 5-1 Unidirectional replication topology configuration

The primary server is logically a data sharing group (D8GG) involving two LPARs, SC53 (host name WTSC53.pok.ibm.com and IP address 9.12.6.77) and SC67, that are part of a two-member data sharing group. The source database is D8G1 on LPAR SC53 and D8G2 on LPAR SC67. One queue manager is defined on each LPAR, both having the same name ‘MQZ1’. Therefore, the queue manager may only be active on one of the LPARs at a given point in time. Any attempt to start both queue managers simultaneously will result in the CSQY003I and CSQY9023E error messages shown in Example 5-1.

Example 5-1 Queue manager MQZ1 startup error message in the MVS log

```
000290  =MQZ1 START QMGR
000090  CSQY003I =MQZ1 QUEUE MANAGER IS ALREADY ACTIVE
000090  CSQ9023E =MQZ1 CSQYSCMD 'START QMGR' ABNORMAL COMPLETION
```

The Q Capture process is started on LPAR SC53. The Q replication queue objects such as the send queue (SENDQ), transmit queue (XMITQ), restart queue (RESTARTQ), and admin queue (ADMINQ) are defined in the shared disk.

The secondary server SC59 has the host name WTSC59.pok.ibm.com and IP address 9.12.4.10. The target database is DB8A, and the queue manager is MQ59. The Q Apply process is started on this server. The Q replication queue objects such as the receive queue (RECVQ) are defined on this server.

Attention: A dynamic virtual IP address (DVIPA) is used in the definition of the channels between the secondary server and the primary server. By providing IP addresses for the z/OS applications that are not associated with a specific physical network attachment or gateway, a Virtual IP Address (VIPA) enables fault tolerance against outages in the IP interfaces on the z/OS host. Dynamically activated VIPAs are called DVIPAs. Using a DVIPA eliminates the need to modify the channel definition's IP address after LPAR SC53 fails, and Q Capture is started on LPAR 67 in order to continue normal replication operations. For more details on dynamic virtual IP address, refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance,* SG24-6517.

The tables used in this scenario are identical to the ones used in Chapter 4, "WebSphere MQ shared queues and unidirectional replication" on page 103, as shown in Figure 4-2 on page 106. The DDL used in creating these tables is shown in Example 4-1 on page 106 and Example 4-2 on page 107, but the schema is ITSOD for shared disk instead of ITSOQ for shared queues.

Figure 5-2 provides a high-level overview of the various objects involved in implementing a unidirectional replication topology for AvantGarde. In Figure 5-2, there appear to be two sets of transmission queues, and sender and receiver channels on each server. However, there is only one set on each server, as can be deduced from the identical names. Figure 5-2 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

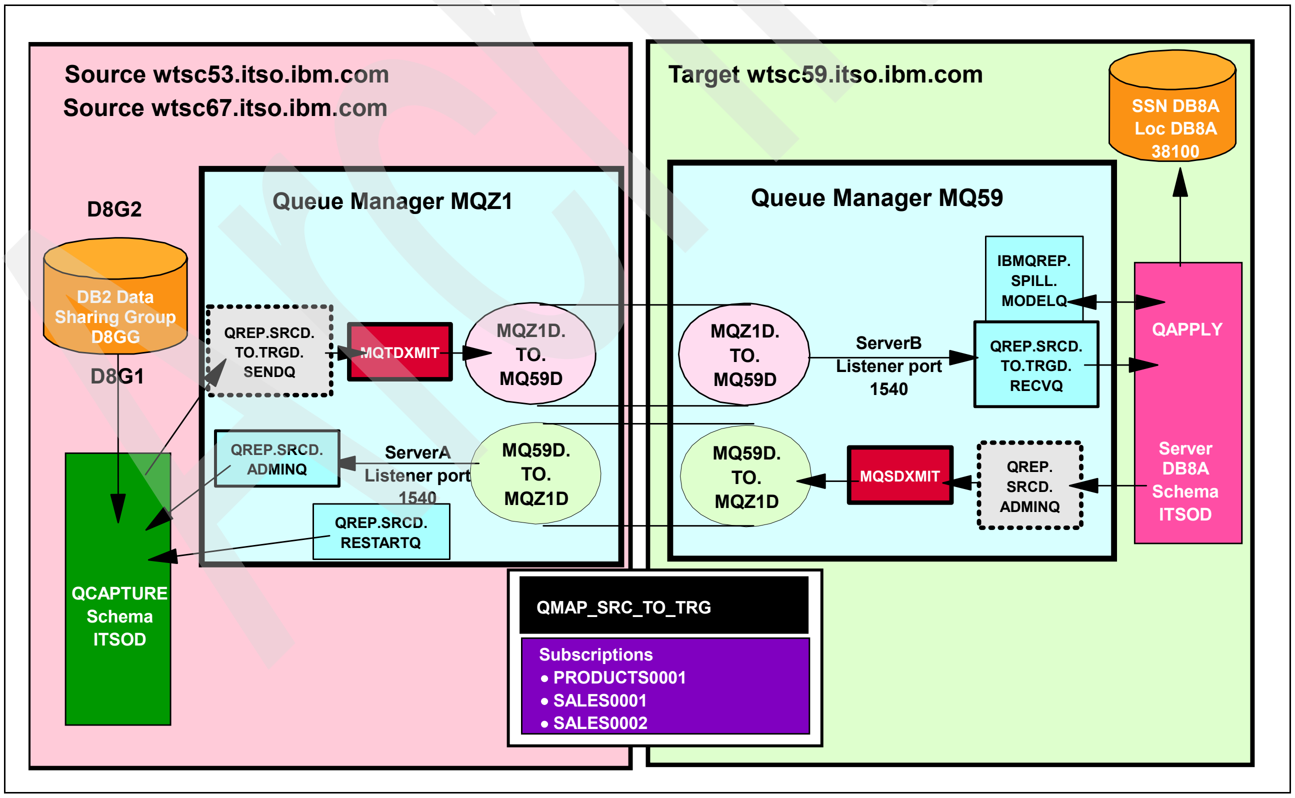


Figure 5-2 Unidirectional replication topology objects overview

5.5 Set up of this environment

Figure 5-3 provides an overview of the steps involved in setting up the unidirectional Q replication environment described in Figure 5-1, Figure 4-2 on page 106, and Figure 5-2. The scripts used can be downloaded from the IBM Redbooks Web site:

ftp://www.redbooks.ibm.com/redbooks/SG247215/

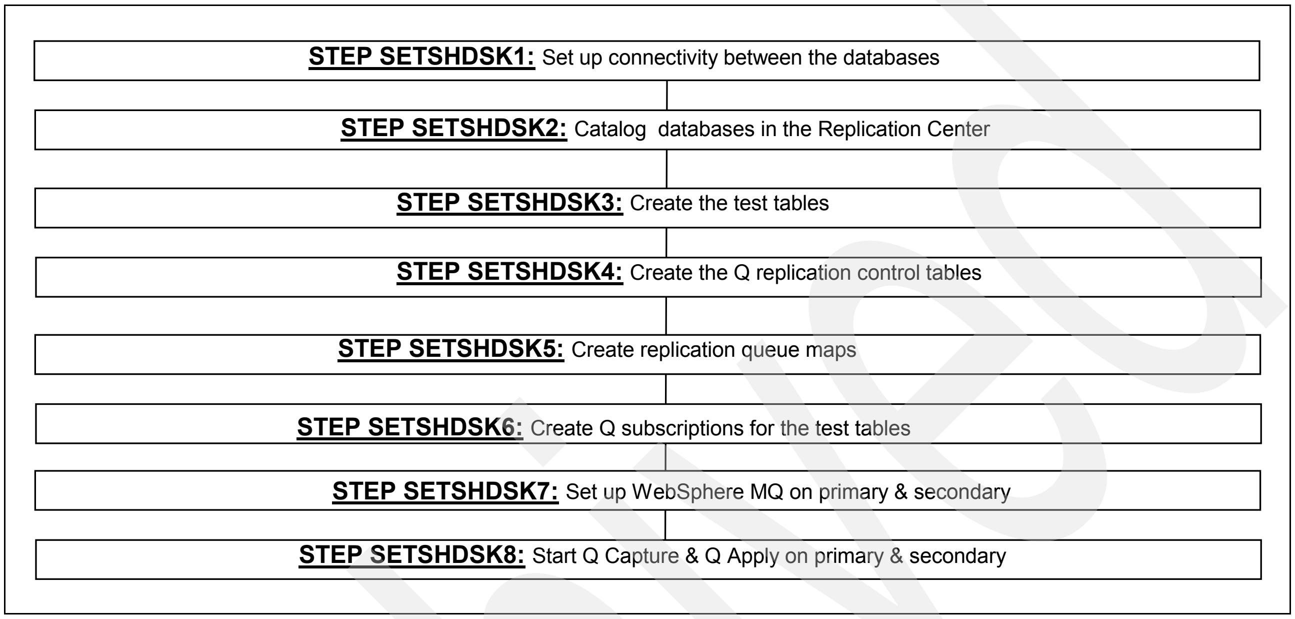

STEP SETSHDSK1: Set up connectivity between the databases
STEP SETSHDSK2: Catalog databases in the Replication Center
STEP SETSHDSK3: Create the test tables
STEP SETSHDSK4: Create the Q replication control tables
STEP SETSHDSK5: Create replication queue maps
STEP SETSHDSK6: Create Q subscriptions for the test tables
STEP SETSHDSK7: Set up WebSphere MQ on primary & secondary
STEP SETSHDSK8: Start Q Capture & Q Apply on primary & secondary

Figure 5-3 Setup of the WebSphere shared disk unidirectional environment

Important: The steps described in Figure 5-3 assume that the shared disk environment has already been set up. Appendix C.2, “WebSphere shared disk overview” on page 255, provides an overview of shared disk.

Each of the steps described in Figure 5-3 is almost identical with those described in Figure 4-4 for setting up the shared disk environment in Chapter 4, “WebSphere MQ shared queues and unidirectional replication” on page 103, with some slight differences, as follows.

5.5.1 STEP SETSHDSK1: Set up connectivity between the databases

In this step connectivity is established between the source and target databases. This involves inserting rows into the SYSIBM.LOCATIONS, SYSIBM.IPNAMES, and SYSIBM.USERNAMES tables on the primary server SC53, as shown in Example 4-6 on page 113, and secondary server SC59, as shown in Example 4-7 on page 114.

5.5.2 STEP SETSHDSK2: Catalog databases in the Replication Center

To configure the Q replication environment from the Replication Center, you need to catalog the source and target databases, as shown in Example 4-8 on page 114.

5.5.3 STEP SETSHDSK3: Create the test tables

The test tables used in the scenario are identical to those described in Example 4-2 on page 107, and are created using DDL, shown in Example 4-1 on page 106 and Example 4-2 on page 107.

5.5.4 STEP SETSHDSK4: Create the Q replication control tables

The DDL for creating the Q replication control tables on the primary server SC53 and secondary server SC59, respectively, are identical to those used in Example 4-9 on page 115 and Example 4-10 on page 119.

5.5.5 STEP SETSHDSK5: Create the replication queue maps

In this step, create the replication queue maps using the SQL generated by the Replication Center, as shown in Example 4-11 on page 124 and Example 4-12 on page 124.

5.5.6 STEP SETSHDSK6: Create Q subscriptions for the test tables

The subscriptions for the test tables are identical to those defined using the SQL generated by the Replication Center, as shown in Example 4-13 on page 125 through Example 4-14 on page 126.

For each of the subscriptions defined, the column HAS_LOADPHASE has a value of 'I' (automatic load), STATE has a value of 'N' (new subscription), and CHANGED_COLS_ONLY has a value of 'Y' (changed columns only for conflict rule).

5.5.7 STEP SETSHDSK7: Set up WebSphere MQ on primary and secondary

Example 5-2 shows the definition of the WebSphere MQ objects on the queue manager MQZ1 on the primary server SC53.

Example 5-3 shows the definition of the WebSphere MQ objects on the queue manager MQ59 on the secondary server SC59.

Example 5-2 JCL to define WebSphere MQ objects on the primary server SC53

```
//ASNQDEF  JOB NOTIFY=&SYSUID,
//         MSGCLASS=H,MSGLEVEL=(1,1),
//         REGION=0M,TIME=NOLIMIT
/*JOBPARM S=SC53
//ASNQMQD1 EXEC PGM=CSQUTIL,PARM='MQZ1'
//STEPLIB   DD DSN=MQZ1.USERAUTH,DISP=SHR
//          DD DSN=MQ600.SCSQANLE,DISP=SHR
//          DD DSN=MQ600.SCSQAUTH,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *

*
* Following are the definitions which are required to send data
* from the SRC capture instance to the TRG apply instance.
* Similar definitions are required for each capture/apply pair.
*
```

```
*

DEFINE REPLACE                                                   +
       QLOCAL(QREP.SRCD.ADMINQ)                                  +
       DESCR('LOCAL DEFN OF ADMINQ FOR SRCD CAPTURE')            +
       PUT(ENABLED)                                              +
       GET(ENABLED)                                              +
       SHARE                                                     +
       DEFSOPT(SHARED)                                           +
       DEFPSIST(YES)

DEFINE REPLACE                                                   +
       QLOCAL(QREP.SRCD.RESTARTQ)                                +
       DESCR('LOCAL DEFN OF RESTART FOR SRCD CAPTURE')           +
       PUT(ENABLED)                                              +
       GET(ENABLED)                                              +
       SHARE                                                     +
       DEFSOPT(SHARED)                                           +
       DEFPSIST(YES)

DEFINE REPLACE                                                   +
       QREMOTE(QREP.SRCD.TO.TRGD.SENDQ)                          +
       DESCR('REMOTE DEFN OF SEND QUEUE FROM SRCD TO TRGD')      +
       PUT(ENABLED)                                              +
       XMITQ(MQTDXMIT)                                           +
       RNAME(QREP.SRCD.TO.TRGD.RECVQ)                            +
       RQMNAME(MQ59)                                             +
       DEFPSIST(YES)


* Following are the definitions which are required to send data
* from an MQSeries queue manager named MQZ1 to a Queue manager
* named MQ59.  These definitions are required only once.
*

DEFINE REPLACE                                                   +
       QLOCAL(MQTDXMIT)                                          +
       DESCR('TRANSMISSION QUEUE TO MQ59')                       +
       USAGE(XMITQ)                                              +
       PUT(ENABLED)                                              +
       GET(ENABLED)                                              +
       TRIGGER                                                   +
       TRIGTYPE(FIRST)                                           +
       TRIGDATA(MQZ1D.TO.MQ59D)                                  +
       INITQ(SYSTEM.CHANNEL.INITQ)

*
* Note that a listener must be started, on the other queue
* manager, for the port number identified in the channel
* sender definition.
*

DEFINE REPLACE                                                   +
       CHANNEL(MQZ1D.TO.MQ59D)                                   +
       CHLTYPE(SDR)                                              +
       TRPTYPE(TCP)                                              +
       DESCR('SENDER CHANNEL TO MQ59')                           +
       XMITQ(MQTDXMIT)                                           +
       CONNAME('9.12.4.10(1540)')
```

```
DEFINE REPLACE                                                  +
       CHANNEL(MQ59D.TO.MQZ1D)                                  +
       CHLTYPE(RCVR)                                            +
       TRPTYPE(TCP)                                             +
       DESCR('RECEIVER CHANNEL FROM MQ59')

DISPLAY QUEUE(*) ALL
```

Example 5-3 JCL to define WebSphere MQ objects on the secondary server SC59

```
//ASNQDEF JOB (POK,999),MSGLEVEL=(1,1),MSGCLASS=H,
//  CLASS=A,NOTIFY=&SYSUID,REGION=0M
//ASNQMQD1 EXEC PGM=CSQUTIL,PARM='MQ59'
//STEPLIB   DD DSN=MQ59.USERAUTH,DISP=SHR
//          DD DSN=MQ600.SCSQANLE,DISP=SHR
//          DD DSN=MQ600.SCSQAUTH,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *



* Following are the definitions which are required to send data
* from an MQSeries queue manager named MQ59 to a Queue manager
* named MQZ1.
*

DEFINE REPLACE                                                  +
       QLOCAL(MQSDXMIT)                                         +
       DESCR('TRANSMISSION QUEUE TO MQZ1')                      +
       USAGE(XMITQ)                                             +
       PUT(ENABLED)                                             +
       GET(ENABLED)                                             +
       TRIGGER                                                  +
       TRIGTYPE(FIRST)                                          +
       TRIGDATA(MQ59D.TO.MQZ1D)                                 +
       INITQ(SYSTEM.CHANNEL.INITQ)

*
* Note that a listener must be started, on the other queue
* manager, for the port number identified in the channel
* sender definition. 9.12.6.11 is a DVIPA for both db2
* subsystems.

DEFINE REPLACE                                                  +
       CHANNEL(MQ59D.TO.MQZ1D)                                  +
       CHLTYPE(SDR)                                             +
       TRPTYPE(TCP)                                             +
       DESCR('SENDER CHANNEL TO MQZ1')                          +
       XMITQ(MQSDXMIT)                                          +
       CONNAME('9.12.6.11(1540)')

DEFINE REPLACE                                                  +
       CHANNEL(MQZ1D.TO.MQ59D)                                  +
       CHLTYPE(RCVR)                                            +
       TRPTYPE(TCP)                                             +
       DESCR('RECEIVER CHANNEL FROM MQZ1')
```

```
*
* Following are the definitions which are required to receive data
* at the TRG apply instance from the SRC capture instance.
*

DEFINE REPLACE                                                      +
       QMODEL('IBMQREP.SPILL.MODELQ')                               +
       DEFSOPT(SHARED)                                              +
       MAXDEPTH(500000)                                             +
       MSGDLVSQ(FIFO)                                               +
       DEFTYPE(PERMDYN)

DEFINE REPLACE                                                      +
       QREMOTE(QREP.SRCD.ADMINQ)                                    +
       DESCR('REMOTE DEFN OF ADMINQ FOR SRCD CAPTURE')              +
       PUT(ENABLED)                                                 +
       XMITQ(MQSDXMIT)                                              +
       RNAME(QREP.SRCD.ADMINQ)                                      +
       RQMNAME(MQZ1)                                                +
       DEFPSIST(YES)

DEFINE REPLACE                                                      +
       QLOCAL(QREP.SRCD.TO.TRGD.RECVQ)                              +
       DESCR('LOCAL RECEIVE QUEUE - SRCD TO TRGD')                  +
       PUT(ENABLED)                                                 +
       GET(ENABLED)                                                 +
       SHARE                                                        +
       DEFSOPT(SHARED)                                              +
       DEFPSIST(YES)                                                +
       INDXTYPE(MSGID)


DISPLAY QUEUE(*) ALL
```

5.5.8 STEP SETSHDSK8: Start Q Capture and Q Apply on primary and secondary

Start Q Capture on the primary server SC53 using JCL, similar to that shown in Example 4-20 on page 131, and Q Apply on the secondary server SC59 using JCL, similar to that shown in Example 4-21 on page 133, after ensuring that the channels are running and the transmit and receive queues are empty. Their successful initialization can be determined by viewing the contents of their respective logs, as described in Example 3-16 on page 50 and Example 3-12 on page 48.

Note: Since the subscriptions were defined with column STATE with a value of ‘N’ in the IBMQREP_SUBS table, the subscriptions are activated when Q Capture starts. The HAS_LOADPHASE value of ‘I’ causes an automatic load to be initiated. You can verify that replication is operating normally as described in “STEP SWUFOPA17: Verify that Q replication up and running” on page 70.

5.6 Test cases

To verify the seamless and synergistic relationship between WebSphere MQ shared disk and Q replication, we executed the following test cases in this configuration:

- Failover operations with no inflight automatic load in progress at the time of failure where unidirectional Q replication is restored to the secondary server SC59 without any data loss after the failure of LPAR SC53. The Q Capture process that was running on the failed LPAR SC53 is started (manually or automatically) on LPAR SC67.
- Failover operations with inflight automatic load in progress at the time of failure where unidirectional Q replication is restored to the secondary server SC59 without any data loss after the failure of LPAR SC53. The Q Capture process that was running on the failed LPAR SC53 is started (manually or automatically) on LPAR SC67.

Attention: The objectives of these test cases and most of the steps listed for each test case are identical to those conducted in Section 4.6, “Test cases” on page 134, for the shared queues scenario. The differences are as follows:

- The shared disk scenarios described here have additional steps to stop the queue manager MQZ1 on LPAR SC53 and start queue manager MQZ1 again on LPAR SC67 after the simulation of the failure of LPAR SC53 through a STOP DB2 MODE(FORCE) command.
- Where the steps are identical, the shared disk scenario steps refer back to their corresponding steps in the shared queues test cases. It should be noted, however, that the shared disk steps use the schema ITSOD instead of ITSOQ that is used in the shared queues steps, and the character ‘D’ is added in various object names for shared disk. For example:
  - CREATE TABLE ITSOD.PRODUCTS in shared disk versus CREATE TABLE ITSOQ.PRODUCTS in shared queues
  - CREATE TABLE ITSOD.IBMQREP_CAPPARMS in shared disk versus CREATE TABLE ITSOQ.IBMQREP_CAPPARMS in shared queues
  - QREP.SRCD.ADMINQ in shared disk versus QREP.SRC.ADMINQ in shared queues

These test cases and results are described in detail in the following sections.

Attention: As mentioned earlier, even though WebSphere queues used by Q replication objects are stored on shared disk, they may only be accessed by queue manager MQZ1 on LPAR SC53 or queue manager MQZ1 on LPAR SC67 in a mutually exclusive manner. Q replication also accesses these queues on shared disk from only one LPAR or the other—in other words, mutually exclusive access to shared Q replication WebSphere MQ objects on the shared disk. Any attempt to start Q Capture on LPAR SC67 when it is already running on LPAR SC53 in our scenario results in an error message `ASN0554E “Q Capture” : “ITSOQ” : “Initial” : The program is already active`, as shown in the MVS log output in Example 4-22 on page 134.

5.6.1 Failover operations with no inflight automatic load in progress

The objective of this test case is to demonstrate that under failover operations with no inflight automatic load in progress, Q replication operates in a seamless fashion in our WebSphere MQ shared disk high availability primary server environment with no data loss.

The following assumptions are made about the test case environment:

- Application workloads are running against both members of the DB2 data sharing group D8GG comprising member databases D8G1 on LPAR SC53 and D8G2 on LPAR SC67.
- Q Capture is running on the primary server LPAR SC53.

- Q Apply is running on the secondary server LPAR SC59.
- The PRODUCTS table is replicated in its entirety to the secondary server.
- The source SALES table is replicated to a SALES_EAST and a SALES_WEST table on the target database depending on the REGION_CODE value on the source table. The REGION_CODE column is not replicated.

Figure 5-4 shows the steps executed in this test case.

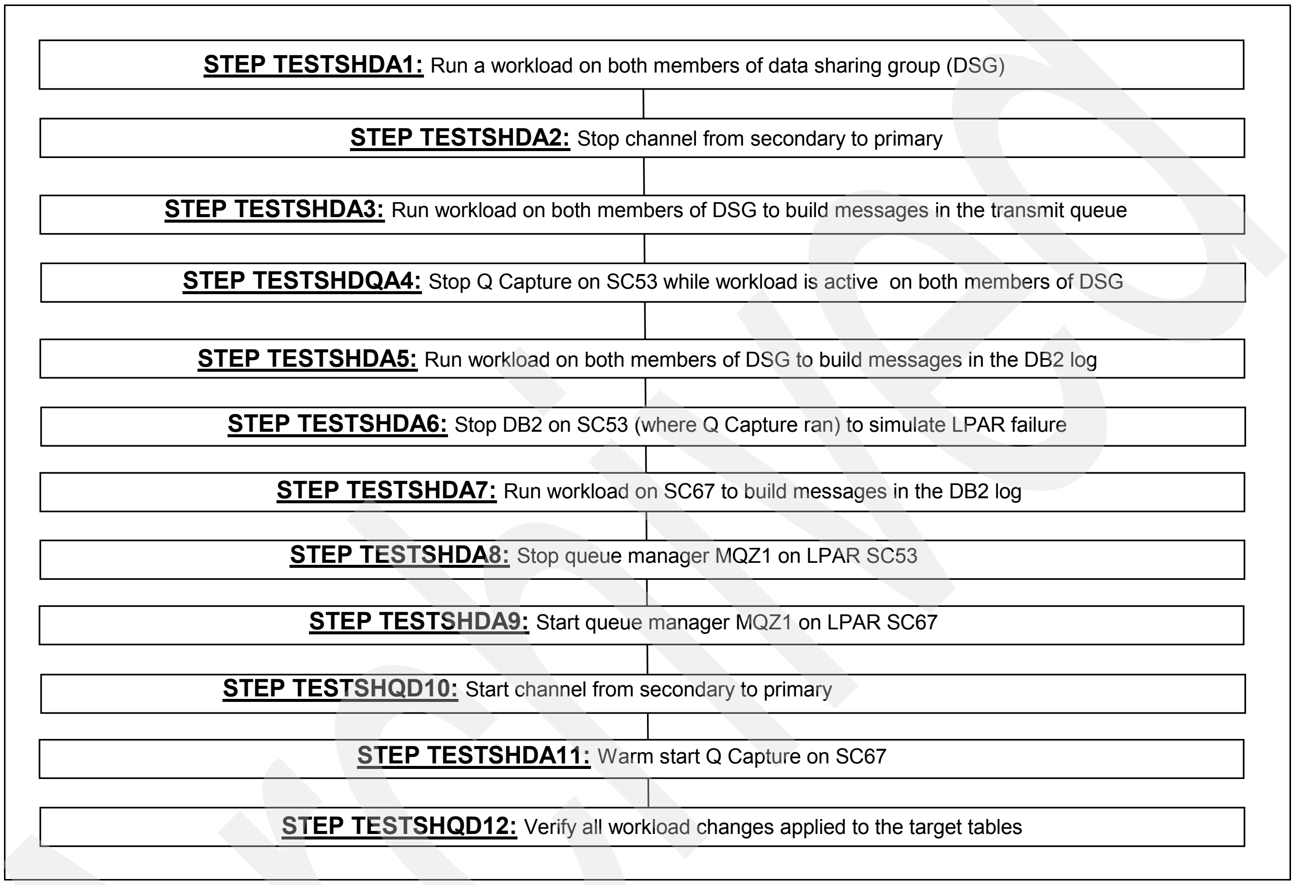

STEP TESTSHDA1: Run a workload on both members of data sharing group (DSG)
STEP TESTSHDA2: Stop channel from secondary to primary
STEP TESTSHDA3: Run workload on both members of DSG to build messages in the transmit queue
STEP TESTSHDQA4: Stop Q Capture on SC53 while workload is active on both members of DSG
STEP TESTSHDA5: Run workload on both members of DSG to build messages in the DB2 log
STEP TESTSHDA6: Stop DB2 on SC53 (where Q Capture ran) to simulate LPAR failure
STEP TESTSHDA7: Run workload on SC67 to build messages in the DB2 log
STEP TESTSHDA8: Stop queue manager MQZ1 on LPAR SC53
STEP TESTSHDA9: Start queue manager MQZ1 on LPAR SC67
STEP TESTSHQD10: Start channel from secondary to primary
STEP TESTSHDA11: Warm start Q Capture on SC67
STEP TESTSHQD12: Verify all workload changes applied to the target tables

Figure 5-4 Failover operations with no inflight automatic load in progress test case with the shared disk

Important: The steps described in Figure 5-3 assume that the prerequisites for a shared disk environment have already been set up. Appendix C.2, “WebSphere shared disk overview” on page 255, provides an overview of shared disk functionality.

Each of the steps described in Figure 5-3 is almost identical with those described in Figure 4-5 on page 136 for setting up the shared disk environment in Chapter 4, “WebSphere MQ shared queues and unidirectional replication” on page 103, with some slight differences, as follows.

STEP TESTSHDA1: Run workload on both members of DSG

Example 4-24 on page 136 shows the SQL used to simulate the workload running on both members of the data sharing group—database D8G1 on LPAR SC53 and D8G2 on LPAR SC67. This is the same workload used in “STEP TESTSHQA1: Run workload on all members of data sharing group” on page 136 for the test case in Section 5.6.1, “Failover operations with no inflight automatic load in progress” on page 174.

The contents of the tables on the primary server SC53 and secondary server are shown in Example 4-25 on page 138 and Example 4-26 on page 139, respectively. The content is identical on the primary server and secondary server indicating successful replication with no data loss.

STEP TESTSHDA2: Stop channel from secondary to primary

In this step, stop the channel between the secondary server SC59 and the primary server SC53, as shown in Figure 4-6 on page 140 through Figure 4-9 on page 141, in order to be able to build changes in the transmit queue in the coupling facility for processing by LPAR SC67 when Q Capture is started on that LPAR.

STEP TESTSHDA3: Run workload on both members of DSG

In this step, run another workload, as shown in Example 4-27 on page 141, on both members of the data sharing group to generate messages in the transmit queue in the shared disk. This is the same workload used in "STEP TESTSHQA3: Run workload on all members of data sharing group" on page 141 for the test case in Section 5.6.1, "Failover operations with no inflight automatic load in progress" on page 174. Figure 4-10 on page 143 and Figure 4-11 on page 143 show the build up of messages (current queue depth of 11) in the transmit queue in the shared disk.

STEP TESTSHQD4: Stop Q Capture on SC53 while workload is active

In this step, stop Q Capture on LPAR SC53, as shown in Example 4-28 on page 143, in order to be able to build changes in the DB2 logs for processing by Q Capture when it is started on LPAR SC67. Its successful stopping can be determined by viewing the contents of the Q Capture log similar to that shown in "Stop Q Capture log contents on the secondary server SC59" on page 43.

STEP TESTSHDA5: Run workload on both members of DSG

In this step, run another workload (as shown in Example 4-29 on page 144) on both members of the data sharing group to generate messages in the DB2 log for Q Capture to process when it is started on LPAR SC67. This is the same workload used in "STEP TESTSHQA5: Run workload on all members of data sharing group" on page 144 for the test case in Section 5.6.1, "Failover operations with no inflight automatic load in progress" on page 174.

STEP TESTSHDA6: Stop DB2 on SC53 to simulate an LPAR failure

In this step, stop DB2 on LPAR SC53 to simulate its failure, as shown in Example 4-30 on page 145.

STEP TESTSHDA7: Run workload on SC67

In this step, run another workload, as shown in Example 4-30 on page 145, on LPAR SC67 only (since LPAR SC53 had failed) to generate messages in the DB2 log for Q Capture to process when it is started on LPAR SC67. This is the same workload used in "STEP TESTSHQA7: Run workload on SC67 to build messages in log" on page 145 for the test case Section 5.6.1, "Failover operations with no inflight automatic load in progress" on page 174.

STEP TESTSHDA8: Stop queue manager MQZ1 on LPAR SC53

In this step, stop the queue manager MQZ1 on LPAR SC53 (assuming that the LPAR is still up and running and only DB2 had failed) by stopping the MQ channel queue initiator and the MQ queue manager, as shown in Example 5-4 and Example 5-5, respectively.

Example 5-4 Stop MQ channel initiator on LPAR SC53

```
=MQZ1 STOP CHINIT
```

```
CSQM137I =MQZ1 CSQMTCHI  STOP CHINIT COMMAND ACCEPTED
+CSQX234I =MQZ1 CSQXLSTT Listener stopped, TRPTYPE=TCP INDISP=GROUP
+CSQX208E =MQZ1 CSQXRESP Error receiving data, 229
 channel MQ59D.TO.MQZ1D,
 connection (9.12.4.10)
 (queue manager MQ59)
 TRPTYPE=TCP RC=00000480 reason=00000000
+CSQX599E =MQZ1 CSQXRESP Channel MQ59D.TO.MQZ1D ended abnormally
CSQM053E =MQZ1 CSQMPCRT Shared channel recovery 231
terminated, DB2 not available
+CSQX411I =MQZ1 CSQXREPO Repository manager stopped
+CSQY204I =MQZ1 CSQXJST ARM DEREGISTER for element SYSMQCHMQZ1 type
233
 SYSMQCH successful
CSQ9022I =MQZ1 CSQXCRPS ' STOP CHINIT' NORMAL COMPLETION
```

Example 5-5   Stop MQ queue manager MQZ1 on LPAR SC53

```
=MQZ1 STOP QMGR
CSQY009I =MQZ1 'STOP QMGR' COMMAND ACCEPTED FROM 243
USER(HUEYKIM), STOP MODE(QUIESCE)
CSQY204I =MQZ1  CSQYSCMA ARM DEREGISTER for element SYSMQMGRMQZ1 type
244
SYSMQMGR successful
=MQZ1 DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED)
CSQY002I =MQZ1 QUEUE MANAGER STOPPING
=MQZ1 DISPLAY USAGE TYPE(ALL)
CSQM297I =MQZ1 CSQMDRTC NO CONN FOUND MATCHING REQUEST CRITERIA
CSQ9022I =MQZ1 CSQMDRTC ' DISPLAY CONN' NORMAL COMPLETION
CSQI010I =MQZ1 Page set usage ... 250
Page Buffer   Total     Unused  Persistent Nonpersistent  Expansion
set  pool     pages      pages  data pages    data pages      count
_  0   0       1078       1036          42             0  USER    0
_  1   0       1078       1066          12             0  USER    0
_  2   1       1078       1078           0             0  USER    0
_  3   2       1078       1077           0             1  USER    0
_  4   3       4318        633        3685             0  USER    0
 End of page set report
CSQP001I =MQZ1 Buffer pool 0 has 50000 buffers
CSQP001I =MQZ1 Buffer pool 1 has 20000 buffers
CSQP001I =MQZ1 Buffer pool 2 has 50000 buffers
CSQP001I =MQZ1 Buffer pool 3 has 20000 buffers
CSQI024I =MQZ1 CSQIDUSE Restart RBA for system as 255
configured=000000000000
CSQI070I =MQZ1 Data set usage ... 256
Data set  RBA/LRSN      DSName
Log, oldest with active unit of work:
_         00003313B526  MQZ1.LOGCOPY1.DS01
Log, oldest for page set recovery:
_         0000330CD0E2  MQZ1.LOGCOPY1.DS01
Log, oldest for CF structure recovery:
_         000000000000  Not found
 End of data set report
CSQ9022I =MQZ1 CSQIDUSE ' DISPLAY USAGE' NORMAL COMPLETION
ATR169I RRS HAS UNSET EXITS FOR RESOURCE MANAGER 258
CSQ.RRSATF.IBM.MQZ1 REASON: UNREGISTERED
CSQM051I =MQZ1 CSQMIGQA Intra-group queuing agent stopping
CSQE006I =MQZ1 Structure APPLICATION1 connection name 260
CSQEMQZGMQZ101 disconnected
CSQP018I =MQZ1 CSQPBCKW CHECKPOINT STARTED FOR ALL BUFFER POOLS
```

```
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 262
BUFFER POOL 1, 0 PAGES WRITTEN
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 263
BUFFER POOL 2, 2 PAGES WRITTEN
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 264
BUFFER POOL 3, 10 PAGES WRITTEN
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 265
BUFFER POOL 0, 21 PAGES WRITTEN
CSQP021I =MQZ1 Page set 0 new media recovery 266
RBA=00003313B526, checkpoint RBA=00003313B526
CSQP021I =MQZ1 Page set 1 new media recovery 267
RBA=00003313B526, checkpoint RBA=00003313B526
CSQP021I =MQZ1 Page set 2 new media recovery 268
RBA=00003313B526, checkpoint RBA=00003313B526
CSQP021I =MQZ1 Page set 3 new media recovery 269
RBA=00003313B526, checkpoint RBA=00003313B526
CSQP021I =MQZ1 Page set 4 new media recovery 270
RBA=00003313B526, checkpoint RBA=00003313B526
CSQ5022I =MQZ1 CSQ5DISC Pending connection to DB2 271
using D8GG ended, queue manager terminating
CSQE006I =MQZ1 Structure CSQ_ADMIN connection name 272
CSQEMQZGMQZ101 disconnected
CSQ9022I =MQZ1 CSQYASCP 'STOP QMGR' NORMAL COMPLETION
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-MQZ1MSTR PROCSTEP MQZ1MSTR    00   2191    .10    .02   50.9  84403
 0      0      0      0     0
IEF404I MQZ1MSTR - ENDED - TIME=12.09.33 - ASID=00B4 - SC53
-MQZ1MSTR ENDED.  NAME-                     TOTAL CPU TIME=   .10
TOTAL ELAPSED TIME=  50.9
IEF352I ADDRESS SPACE UNAVAILABLE
$HASP395 MQZ1MSTR ENDED
IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=00B4.
CSQ3104I =MQZ1    CSQ3ECOX - TERMINATION COMPLETE
CSQ3100I =MQZ1    CSQ3ECOX - SUBSYSTEM MQZ1 READY FOR START COMMAND
```

STEP TESTSHDA9: Start queue manager MQZ1 on LPAR SC67

In this step, start the queue manager MQZ1 on LPAR SC67 by starting the MQ queue manager and the MQ channel queue initiator, as shown in Example 5-6 and Example 5-7, respectively.

Example 5-6 Start MQ queue manager MQZ1 on LPAR SC67

```
=MQZ1 START QMGR
S MQZ1MSTR
$HASP100 MQZ1MSTR ON STCINRDR
IEF695I START MQZ1MSTR WITH JOBNAME MQZ1MSTR IS ASSIGNED TO USER
MQZ1MSTR, GROUP SYS1
$HASP373 MQZ1MSTR STARTED
IEF403I MQZ1MSTR - STARTED - TIME=12.11.27 - ASID=00DE - SC67
CSQY000I =MQZ1 IBM WebSphere MQ for z/OS V6
CSQY001I =MQZ1 QUEUE MANAGER STARTING, USING PARAMETER MODULE CSQZPARM
CSQ3111I =MQZ1 CSQYSCMD - EARLY PROCESSING PROGRAM IS V6 LEVEL 003-002
CSQY100I =MQZ1 SYSTEM parameters ...
CSQY101I =MQZ1 CTHREAD=1300, IDBACK=600, IDFORE=600, LOGLOAD=500000
CSQY102I =MQZ1 CMDUSER=CSQOPR, QMCCSID=0, ROUTCDE=( 1)
CSQY103I =MQZ1 SMFACCT=NO (00000000), SMFSTAT=NO (00000000),
STATIME=30
CSQY104I =MQZ1 OTMACON= 773
(        ,                ,DFSYDRU0,2147483647,CSQ)
CSQY105I =MQZ1 TRACSTR=( 1), TRACTBL=99
CSQY106I =MQZ1 EXITTCB=8, EXITLIM=30, WLMTIME=30, WLMTIMU=MINS
CSQY107I =MQZ1 QSGDATA=(MQZG,DB8GU,D8GG,10,4)
CSQY108I =MQZ1 RESAUDIT=YES, QINDXBLD=WAIT, CLCACHE=STATIC
CSQY110I =MQZ1 LOG parameters ...
CSQY111I =MQZ1 INBUFF=60, OUTBUFF=4000, MAXRTU=2, MAXARCH=500
CSQY112I =MQZ1 TWOACTV=NO, TWOARCH=NO, TWOBSDS=YES
CSQY113I =MQZ1 OFFLOAD=NO, WRTHRSH=15, DEALLCT=0
CSQY120I =MQZ1 ARCHIVE parameters ...
CSQY121I =MQZ1 UNIT=TAPE, UNIT2=, ALCUNIT=BLK, 783
PRIQTY=4320, SECQTY=540, BLKSIZE=28672
CSQY122I =MQZ1 ARCPFX1=CSQARC1, ARCPFX2=CSQARC2, TSTAMP=NO
CSQY123I =MQZ1 ARCRETN=9999, ARCWTOR=YES, ARCWRTC=( 1 ,3 ,4)
CSQY124I =MQZ1 CATALOG=NO, COMPACT=NO, PROTECT=NO, QUIESCE=5
CSQY201I =MQZ1  CSQYSTRT ARM REGISTER for element 787
SYSMQMGRMQZ1 type SYSMQMGR successful
CSQJ127I =MQZ1 SYSTEM TIME STAMP FOR BSDS=2006-03-07 12:09:31.75
CSQJ001I =MQZ1 CURRENT COPY 1 ACTIVE LOG DATA SET IS 789
DSNAME=MQZ1.LOGCOPY1.DS01, STARTRBA=000032A00000 ENDRBA=00003326FFFF
CSQJ099I =MQZ1 LOG RECORDING TO COMMENCE WITH 790
STARTRBA=00003313D000
CSQW130I =MQZ1 'GLOBAL' TRACE STARTED, ASSIGNED TRACE NUMBER 01
CSQ5001I =MQZ1 CSQ5CONN Connected to DB2 D8G2
CSQH021I =MQZ1 CSQHINSQ SUBSYSTEM security switch set 793
OFF, profile 'MQZ1.NO.SUBSYS.SECURITY' found
CSQP007I =MQZ1 Page set 0 uses buffer pool 0
CSQP007I =MQZ1 Page set 1 uses buffer pool 0
CSQP007I =MQZ1 Page set 2 uses buffer pool 1
CSQP007I =MQZ1 Page set 3 uses buffer pool 2
CSQP007I =MQZ1 Page set 4 uses buffer pool 3
CSQY220I =MQZ1 Queue manager is using 123 MB of local 799
storage, 1643 MB are free
CSQV452I =MQZ1 CSQVXLDR Cluster workload exits not available
CSQR001I =MQZ1 RESTART INITIATED
CSQR003I =MQZ1 RESTART - PRIOR CHECKPOINT RBA=00003313B526
CSQR004I =MQZ1 RESTART - UR COUNTS - 803
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0
IXL014I IXLCONN REQUEST FOR STRUCTURE MQZGCSQ_ADMIN 804
WAS SUCCESSFUL.  JOBNAME: MQZ1MSTR ASID: 00DE
CONNECTOR NAME: CSQEMQZGMQZ101 CFNAME: CF03
CSQE005I =MQZ1 Structure CSQ_ADMIN connected as 805
CSQEMQZGMQZ101, version=BD3F9D4C842AEC06 00010216
CSQI049I =MQZ1 Page set 0 has media recovery 806
RBA=00003313B526, checkpoint RBA=00003313B526
CSQI049I =MQZ1 Page set 1 has media recovery 807
RBA=00003313B526, checkpoint RBA=00003313B526
CSQI049I =MQZ1 Page set 2 has media recovery 808
RBA=00003313B526, checkpoint RBA=00003313B526
CSQI049I =MQZ1 Page set 3 has media recovery 809
RBA=00003313B526, checkpoint RBA=00003313B526
CSQI049I =MQZ1 Page set 4 has media recovery 810
RBA=00003313B526, checkpoint RBA=00003313B526
CSQR030I =MQZ1 Forward recovery log range 811
from RBA=00003313B526 to RBA=00003313C1F0
CSQR005I =MQZ1 RESTART - FORWARD RECOVERY COMPLETE - 812
IN COMMIT=0, INDOUBT=0
CSQR032I =MQZ1 Backward recovery log range 813
```

```
from RBA=00003313C1F0 to RBA=00003313C1F0
CSQR006I =MQZ1 RESTART - BACKWARD RECOVERY COMPLETE - 814
INFLIGHT=0, IN BACKOUT=0
CSQR002I =MQZ1 RESTART COMPLETED
CSQP018I =MQZ1 CSQPBCKW CHECKPOINT STARTED FOR ALL BUFFER POOLS
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 817
BUFFER POOL 3, 3 PAGES WRITTEN
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 818
BUFFER POOL 1, 2 PAGES WRITTEN
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 819
BUFFER POOL 2, 2 PAGES WRITTEN
CSQP019I =MQZ1 CSQP1DWP CHECKPOINT COMPLETED FOR 820
BUFFER POOL 0, 31 PAGES WRITTEN
CSQE041E =MQZ1 Structure APPLICATION1 backup is more than a day old
=MQZ1 DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED)
CSQP021I =MQZ1 Page set 0 new media recovery 823
RBA=00003313E0E2, checkpoint RBA=00003313E0E2
CSQP021I =MQZ1 Page set 1 new media recovery 824
RBA=00003313E0E2, checkpoint RBA=00003313E0E2
CSQP021I =MQZ1 Page set 2 new media recovery 825
RBA=00003313E0E2, checkpoint RBA=00003313E0E2
CSQP021I =MQZ1 Page set 3 new media recovery 826
RBA=00003313E0E2, checkpoint RBA=00003313E0E2
CSQP021I =MQZ1 Page set 4 new media recovery 827
RBA=00003313E0E2, checkpoint RBA=00003313E0E2
CSQI007I =MQZ1 CSQIRBLD BUILDING IN-STORAGE INDEX FOR 828
QUEUE QREP.POKB.TO.POKA.RECVQ
CSQM297I =MQZ1 CSQMDRTC NO CONN FOUND MATCHING REQUEST CRITERIA
CSQ9022I =MQZ1 CSQMDRTC ' DISPLAY CONN' NORMAL COMPLETION
CSQI007I =MQZ1 CSQIRBLD BUILDING IN-STORAGE INDEX FOR 830
QUEUE SYSTEM.CHANNEL.SYNCQ
CSQI007I =MQZ1 CSQIRBLD BUILDING IN-STORAGE INDEX FOR QUEUE NEALEQ
CSQI007I =MQZ1 CSQIRBLD BUILDING IN-STORAGE INDEX FOR 833
QUEUE SYSTEM.CLUSTER.REPOSITORY.QUEUE
CSQI006I =MQZ1 CSQIRBLD COMPLETED IN-STORAGE INDEX FOR QUEUE NEALEQ
CSQI006I =MQZ1 CSQIRBLD COMPLETED IN-STORAGE INDEX 835
FOR QUEUE SYSTEM.CLUSTER.REPOSITORY.QUEUE
CSQI006I =MQZ1 CSQIRBLD COMPLETED IN-STORAGE INDEX 836
FOR QUEUE SYSTEM.CHANNEL.SYNCQ
CSQI006I =MQZ1 CSQIRBLD COMPLETED IN-STORAGE INDEX 837
FOR QUEUE QREP.POKB.TO.POKA.RECVQ
IXL014I IXLCONN REQUEST FOR STRUCTURE MQZGAPPLICATION1 838
WAS SUCCESSFUL.  JOBNAME: MQZ1MSTR ASID: 00DE
CONNECTOR NAME: CSQEMQZGMQZ101 CFNAME: CF03
CSQE005I =MQZ1 Structure APPLICATION1 connected as 839
CSQEMQZGMQZ101, version=BD3F9D83D757A564 000100D5
=MQZ1 DISPLAY SYSTEM
=MQZ1 DISPLAY LOG
CSQJ322I =MQZ1 DISPLAY SYSTEM report ... 842
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
CTHREAD     1300
IDBACK      600
IDFORE      600
LOGLOAD     500000
CMDUSER     CSQOPR
QMCCSID     0
ROUTCDE     1
SMFACCT     NO
SMFSTAT     NO
```

```
STATIME     30
OTMACON
  GROUP
  MEMBER
  DRUEXIT   DFSYDRU0
  AGE       2147483647
  TPIPEPFX  CSQ
TRACSTR     1
TRACTBL     99
EXITTCB     8
EXITLIM     30
WLMTIME     30
WLMTIMU     MINS
QSGDATA
  QSGNAME   MQZG
  DSGNAME   DB8GU
  DB2NAME   D8GG
  DB2SERV   10
  DB2BLOB   0
RESAUDIT    YES
QINDXBLD    WAIT
CLCACHE     STATIC
End of SYSTEM report
CSQ9022I =MQZ1 CSQJC001 ' DISPLAY SYSTEM' NORMAL COMPLETION
=MQZ1 DISPLAY ARCHIVE
CSQJ322I =MQZ1 DISPLAY LOG report ... 845
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     4000
MAXRTU      2
MAXARCH     2
TWOACTV     NO
TWOARCH     NO
TWOBSDS     YES
OFFLOAD     NO
WRTHRSH     15
DEALLCT     0
End of LOG report
CSQJ370I =MQZ1 LOG status report ... 846
Copy %Full DSName
 1     86  MQZ1.LOGCOPY1.DS01
 2         Inactive
Restarted at 2006-03-07 12:11:29 using RBA=00003313D000
Latest RBA=00003313EE00
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQ9022I =MQZ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
=MQZ1 DISPLAY USAGE
CSQJ322I =MQZ1 DISPLAY ARCHIVE report ... 849
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
UNIT        TAPE
UNIT2
ALCUNIT     BLK
PRIQTY      4320
SECQTY      540
BLKSIZE     28672
ARCPFX1     CSQARC1
ARCPFX2     CSQARC2
```

```
TSTAMP      NO
ARCRETN     9999
ARCWTOR     YES
ARCWRTC     1 ,3 ,4
CATALOG     NO
COMPACT     NO
PROTECT     NO
QUIESCE     5
End of ARCHIVE report
CSQJ325I =MQZ1 ARCHIVE tape unit report ... 850
Addr St CorrelID VolSer DSName
---- -- -------- ------ --------------------------------------------
No tape archive reading activity
End of tape unit report
CSQ9022I =MQZ1 CSQJC001 ' DISPLAY ARCHIVE' NORMAL COMPLETION
CSQM050I =MQZ1 CSQMIGQA Intra-group queuing agent 852
starting, TCB=0065DA10
CSQI010I =MQZ1 Page set usage ... 853
Page Buffer   Total     Unused  Persistent Nonpersistent  Expansion
set  pool     pages      pages  data pages    data pages      count
_  0   0       1078       1036          42             0  USER    0
_  1   0       1078       1066          12             0  USER    0
_  2   1       1078       1078           0             0  USER    0
_  3   2       1078       1078           0             0  USER    0
_  4   3       4318        633        3685             0  USER    0
 End of page set report
CSQP001I =MQZ1 Buffer pool 0 has 50000 buffers
CSQP001I =MQZ1 Buffer pool 1 has 20000 buffers
CSQP001I =MQZ1 Buffer pool 2 has 50000 buffers
CSQP001I =MQZ1 Buffer pool 3 has 20000 buffers
CSQI024I =MQZ1 CSQIDUSE Restart RBA for system as 858
configured=000000000000
CSQ9022I =MQZ1 CSQIDUSE ' DISPLAY USAGE' NORMAL COMPLETION
CSQY022I =MQZ1 QUEUE MANAGER INITIALIZATION COMPLETE
CSQ9022I =MQZ1 CSQYASCP 'START QMGR' NORMAL COMPLETION
```

Example 5-7 Start MQ channel initiator on LPAR SC67

```
=MQZ1 START CHINIT
S MQZ1CHIN,JOBNAME=MQZ1CHIN
CSQM138I =MQZ1 CSQMSCHI CHANNEL INITIATOR STARTING
$HASP100 MQZ1CHIN ON STCINRDR
IEF695I START MQZ1CHIN WITH JOBNAME MQZ1CHIN IS ASSIGNED TO USER
MQZ1CHIN, GROUP SYS1
$HASP373 MQZ1CHIN STARTED
IEF403I MQZ1CHIN - STARTED - TIME=12.14.37 - ASID=00D9 - SC67
CSQX000I =MQZ1 CSQXJST IBM WebSphere MQ for z/OS V6
CSQX001I =MQZ1 CSQXJST Channel initiator starting
CSQY201I =MQZ1 CSQXJST ARM REGISTER for element SYSMQCHMQZ1 type 885
SYSMQCH successful
CSQX030I =MQZ1 CSQXJST 'GLOBAL' trace started, assigned trace number
00
+CSQX002I =MQZ1 CSQXGIP Queue-sharing group is MQZG
+CSQX070I =MQZ1 CSQXGIP CHINIT parameters ...
+CSQX071I =MQZ1 CSQXGIP CHIADAPS=8, CHIDISPS=5, LSTRTMR=60
+CSQX072I =MQZ1 CSQXGIP MAXCHL=200, ACTCHL=200
+CSQX073I =MQZ1 CSQXGIP CHLEV=ENABLED, SSLEV=ENABLED
+CSQX074I =MQZ1 CSQXGIP MONCHL=OFF, MONACLS=QMGR
+CSQX075I =MQZ1 CSQXGIP ADOPTMCA=YES, ADOPTCHK=ALL
+CSQX078I =MQZ1 CSQXGIP IGQ=DISABLED, CHADEXIT=
+CSQX079I =MQZ1 CSQXGIP TRAXSTR=YES, TRAXTBL=2
+CSQX080I =MQZ1 CSQXGIP SSLTASKS=0, SSLRKEYC=0
+CSQX081I =MQZ1 CSQXGIP SSLKEYR=
+CSQX082I =MQZ1 CSQXGIP SSLCRLNL=
+CSQX085I =MQZ1 CSQXGIP LU62CHL=200, LUGROUP= , LUNAME= , LU62ARM=
+CSQX090I =MQZ1 CSQXGIP TCPCHL=200, TCPKEEP=NO, TCPNAME=TCPIP
+CSQX091I =MQZ1 CSQXGIP TCPSTACK=SINGLE, IPADDRV=IPV4
+CSQX092I =MQZ1 CSQXGIP OPORTMIN=0, OPORTMAX=0
+CSQX093I =MQZ1 CSQXGIP DNSWLM=NO, DNSGROUP=
+CSQX094I =MQZ1 CSQXGIP RCVTIME=0, RCVTTYPE=MULTIPLY, RCVTMIN=0
+CSQX011I =MQZ1 CSQXGIP Client attachment feature available
+CSQX141I =MQZ1 CSQXADPI 8 adapter subtasks started, 0 failed
+CSQX410I =MQZ1 CSQXREPO Repository manager started
+CSQX151I =MQZ1 CSQXSSLI 0 SSL server subtasks started, 0 failed
+CSQX015I =MQZ1 CSQXSPRI 5 dispatchers started, 0 failed
CSQ9022I =MQZ1 CSQXCRPS ' START CHINIT' NORMAL COMPLETION
CSQM052I =MQZ1 CSQMPCRT Shared channel recovery 911
completed for MQZ1, 2 channels found, 0 FIXSHARED, 2 recovered
+CSQX020I =MQZ1 CSQXSPRI Shared channel recovery completed
+CSQX022I =MQZ1 CSQXSUPR Channel initiator initialization complete
+CSQX500I =MQZ1 CSQXRCTL Channel MQZ1.TO.MQS1 started
+CSQX500I =MQZ1 CSQXRCTL Channel MQZ1D.TO.MQ59D started
+CSQX558E =MQZ1 CSQXRCTL Remote channel MQZ1D.TO.MQ59D not available
+CSQX251I =MQZ1 CSQXSTRL Listener started, TRPTYPE=TCP INDISP=GROUP
+CSQX599E =MQZ1 CSQXRCTL Channel MQZ1D.TO.MQ59D ended abnormally
+CSQX023I =MQZ1 CSQXLSTT Listener started, 919
 port 1540 address *,
 TRPTYPE=TCP INDISP=GROUP
+ CSQU012I  CSQUTIL Initialization command handling completed
```

STEP TESTSHDA10: Start channel from secondary to primary

The status at this point is as follows:

- Unpropagated changes exist on the DB2 log on database D8G1, DB2 log on database D8G2, and the transmit queue in the shared disk.
- LPAR SC53 has crashed.
- The channel from the secondary server SC59 to the primary server SC53 is stopped.

Before Q Capture is started on LPAR SC67, start the channel as shown in Figure 4-12 on page 146 through Figure 4-19 on page 149 to enable changes from the transmit queue in the shared disk to propagate right away to the receive queue on the secondary server SC59. Figure 4-20 on page 150 shows the contents of the transmit queue being reduced to zero.

STEP TESTSHDA11: Warm start Q Capture on SC67

In this step, Q Capture is started manually on LPAR SC67 similar to that shown in Example 4-21 on page 133, except that the process is started on LPAR SC67. Its successful initialization can be determined by viewing the contents of the Q Capture log, as described in Example 3-16 on page 50.

STEP TESTSHDA12: Verify all workload changes applied to target tables

With the failover completed, verify that all the workload changes that occurred on the primary server databases D8G1 and D8G2 have been propagated to the target tables on the secondary server SC59. This is achieved by viewing the contents of the tables on the primary server and the secondary server, as shown in Example 4-32 on page 150 and Example 4-33 on page 151, respectively.

Attention: The contents of the tables on the primary server and secondary server are identical, indicating that replication continued seamlessly without any data loss even on the failure of LPAR SC53. The only procedures required were to start the WebSphere MQ queue manager and warm start Q Capture on the other LPAR SC67 either manually or automatically.

5.6.2 Failover operations with inflight automatic load in progress

The objective of this test case is to demonstrate that under failover operations with inflight automatic load in progress, Q replication operates in a seamless fashion in our WebSphere MQ shared disk high availability primary server environment with no data loss. Some additional steps need to be added to cope with inflight automatic loads that failed when the LPAR it was running on crashed.

Attention: If the automatic load happens to be running on a member of a DB2 data sharing group that is different from the one that has Q Capture running, then the crash of the LPAR running Q Capture does *not* impact the normal completion of the automatic load.

The following assumptions are made about the test case environment:

- Application workloads are running against both members of the DB2 data sharing group D8GG comprising member databases D8G1 on LPAR SC53 and D8G2 on LPAR SC67.
- Q Capture is running on the primary server LPAR SC53.
- Q Apply is running on the secondary server LPAR SC59.
- The PRODUCTS table is replicated in its entirety to the secondary server.
- The source SALES table is replicated to a SALES_EAST and a SALES_WEST table on the source database depending on the REGION_CODE value on the source table. The REGION_CODE column is not replicated.

Figure 5-5 shows the steps executed in this test case.

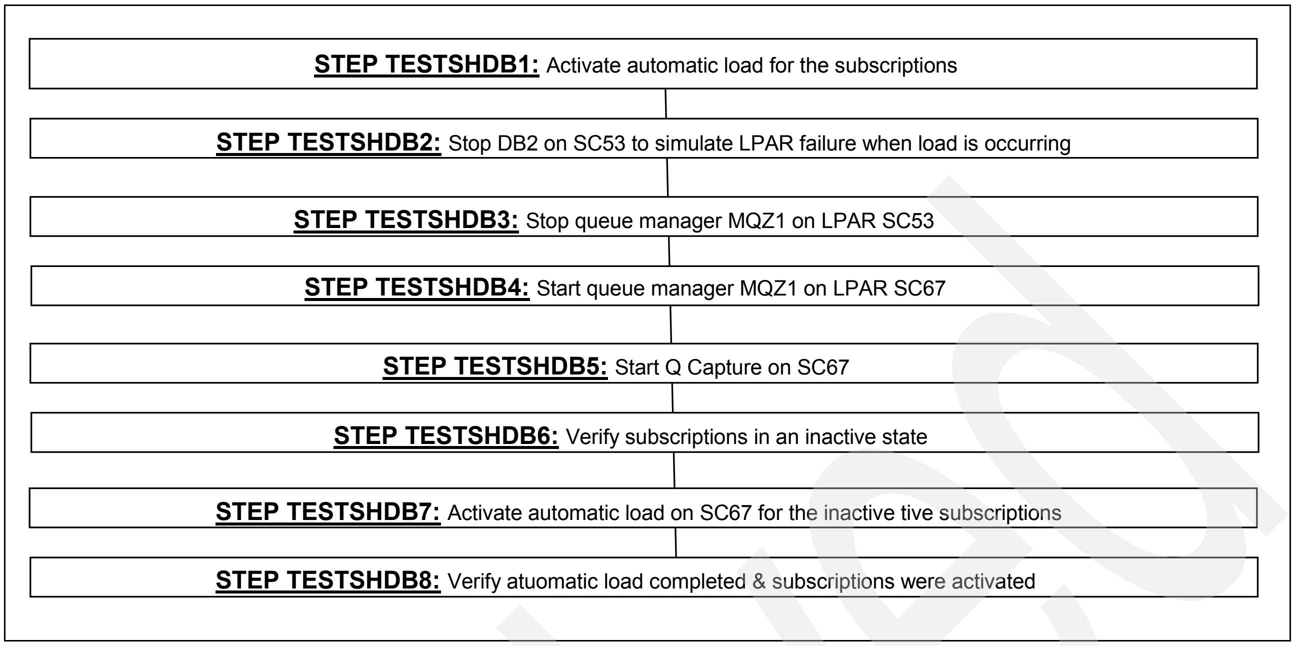

STEP TESTSHDB1: Activate automatic load for the subscriptions
STEP TESTSHDB2: Stop DB2 on SC53 to simulate LPAR failure when load is occurring
STEP TESTSHDB3: Stop queue manager MQZ1 on LPAR SC53
STEP TESTSHDB4: Start queue manager MQZ1 on LPAR SC67
STEP TESTSHDB5: Start Q Capture on SC67
STEP TESTSHDB6: Verify subscriptions in an inactive state
STEP TESTSHDB7: Activate automatic load on SC67 for the inactive tive subscriptions
STEP TESTSHDB8: Verify atuomatic load completed & subscriptions were activated

Figure 5-5 Failover operations w/ inflight automatic load in progress test case with shared disk

Each of the steps shown in Figure 5-5 is described in more detail here.

Attention: We have not included the steps that verify that the content on the primary server and secondary server match since this was demonstrated in Section 5.6.1, "Failover operations with no inflight automatic load in progress" on page 174.

STEP TESTSHDB1: Activate automatic load for the subscription

In this step, activate the PRODUCTS0001 subscription that is inactive, as shown in Figure 4-22 on page 154, and has been defined to be loaded automatically. Over a million rows were loaded into the PRODUCTS table in the primary server SC53 in order to be able to crash LPAR SC53 while the automatic load of the corresponding table on the secondary server SC59 was in progress.

Note: We took certain steps (not documented here) to ensure that the load was running on the same LPAR SC53 where Q Capture was running.

The PRODUCTS0001 subscription was activated by inserting a CAPSTART signal for it in the IBMQREP_SIGNAL table in the D8G1 database on the primary server SC53, as shown in Example 4-34 on page 154.

Figure 4-23 on page 155 shows the PRODUCTS0001 subscription being in the Loading state when automatic load has begun.

STEP TESTSHDB2: Stop DB2 on SC53 to simulate LPAR failure

In this step, stop DB2 on LPAR SC53 to simulate its failure, as shown in Example 4-30 on page 145.

The contents of the Q Capture log on the primary server SC53 shown in Example 4-35 on page 156 indicate that the Q Capture program fails after receiving a connection failure return code since DB2 is down.

The contents of the Q Apply log on the secondary server SC59 shown in Example 4-36 on page 156 indicate that the subscription is deactivated because of the error action option chosen when the load fails as a result of DB2 being unavailable at the primary server SC53.

Example 4-37 on page 157 shows the contents of the load trace file about the failure of the load due to a connection failure.

Example 4-38 on page 158 displays the status of the tablespace in which the PRODUCTS table resides on the secondary server as being in a RECP[1] status after the automatic load failed

Attention: Whenever an automatic load fails, the target tablespace will be placed in a RECP state. One way to remove this state is to initiate the automatic load again.

Figure 4-24 on page 159 shows the state of the PRODUCTS0001 subscription as remaining in the Loading state.

STEP TESTSHDB3: Stop queue manager MQZ1 on LPAR SC53

In this step, stop the queue manager MQZ1 on LPAR SC53 (assuming that the LPAR is still up and running and only DB2 had failed) by stopping the MQ channel queue initiator and the MQ queue manager, as shown in Example 5-4 on page 176 and Example 5-5 on page 177, respectively.

STEP TESTSHDB4: Start queue manager MQZ1 on LPAR SC67

In this step, start the queue manager MQZ1 on LPAR SC67 by starting the MQ queue manager and the MQ channel queue initiator, as shown in Example 5-6 on page 178 and Example 5-7 on page 182, respectively.

STEP TESTSHDB5: Start Q Capture on primary server SC67

In this step, Q Capture is started manually on LPAR SC67, similar to that shown in Example 4-21 on page 133, except that the process is started on LPAR SC67. Its successful initialization can be determined by viewing the contents of the Q Capture log, as described in Example 3-16 on page 50.

STEP TESTSHDB6: Verify subscriptions in an inactive state

Verify that the subscriptions are in an inactive state, as described in “STEP TESTSHQB2: Stop DB2 on SC53 to simulate LPAR failure” on page 155.

STEP TESTSHDB7: Activate automatic load on SC67 for inactive subs

After identifying the subscriptions that were inactivated by the Q Apply program, you must activate them by issuing a CAPSTART signal in the IBMQREP_SIGNAL table in the D8G1 database on the primary server SC67, as shown in Example 4-41 on page 161.

After the CAPSTART signal, the subscription initiates the automatic load. Figure 4-26 on page 162 shows the PRODUCTS0001 subscription being in the loading state using the Replication Center.

STEP TESTSHDB8: Verify automatic load completed and subs activated

In this step, verify the successful completion of the automatic load by viewing the contents of the load trace file, as shown in Example 4-42 on page 162. The active state of this subscription may be verified using the Replication Center (not shown here).

[1] Recovery pending state

A

Summary of code and scripts used in the scenarios

In this appendix we provide a summary of the code and scripts used in the scenarios and that can be downloaded from the IBM Redbooks Web site.

A.1 Summary

The code and scripts used in setting up the scenarios and test cases can be downloaded from the IBM Redbooks Web site at:

ftp://www.redbooks.ibm.com/redbooks/SG247215/

The code and scripts are packaged as follows:

1. BIDI.zip file that includes the code and scripts to set up the environment and perform the steps for controlled failover, switchback from controlled failover, uncontrolled failover, switchback from uncontrolled failover options A, B, C, and D.

   A BIDIREADME.txt file is associated with this zip file that lists its contents with a brief description of each of the following folders. Each of the folders in turn contain another readme.txt file which in turn describes the contents of the folder.

   – BIDISETUP folder
   – CONTROLLEDFAILOVER folder
   – SWITCHBACKCF folder
   – UNCONTROLLEDFAILOVER folder
   – SWITCHBACKUFOPTIONA folder
   – SWITCHBACKUFOPTIONB folder
   – SWITCHBACKUFOPTIONC folder
   – SWITCHBACKUFOPTIOND folder

2. SHQUEUE.zip file that includes the code and scripts to set up the WebSphere MQ shared queues high availability unidirectional Q replication environment, and perform the steps for failover with no inflight automatic load in progress, and failover with inflight automatic load in progress.

   A SHQUEUEREADME.txt file is associated with this zip file that lists its contents with a brief description of each of the following folders. Each of the folders in turn contains another readme.txt file, which in turn describes the contents of the folder.

   – SHQUEUESETUP folder
   – FAILOVERNOINFLIGHT folder
   – FAILOVERWITHINFLIGHT folder

3. SHDISK.zip file that includes the code and scripts to set up the WebSphere MQ shared disk high availability unidirectional Q replication environment, and perform the steps for failover with no inflight automatic load in progress, and failover with inflight automatic load in progress.

   A SHDISKREADME.txt file is associated with this zip file that lists its contents with a brief description of each of the following folders. Each of the folders in turn contains another readme.txt file, which in turn describes the contents of the folder.

   – SHDISKSETUP folder
   – FAILOVERNOINFLIGHT folder
   – FAILOVERWITHINFLIGHT folder

B

Exception processing in a bidirectional Q replication environment

In this appendix we describe how exceptions are created and recorded in a bidirectional Q replication environment. Examples of various bidirectional Q replication scenarios that result in exceptions are provided with a discussion of the corresponding contents of the IBMQREP_EXCEPTIONS table.

The topics covered include:

- Exceptions overview
- Sample exceptions in a bidirectional scenario
- Data inconsistencies in an HA bidirectional scenario

B.1 Exceptions overview

Exceptions are rows that a Q Apply program records in the IBMQREP_EXCEPTIONS table when it encounters unexpected conditions such as conflicts or SQL errors. The Q Apply program saves these rows in XML format, using the same tags and schema[1] that the Q Capture program uses to create an XML data message for event publishing. The rows are saved in XML so that an application can recover the data, and also because the tagging makes it easier to view the column names and values, and the type of row operation such as an insert, update, or delete. Details in these rows stored in the IBMQREP_EXCEPTIONS table (shown in Table B-1) include the time when the error or conflict occurred, the names of the receive queue and Q subscription, the reason for the unexpected condition, and source commit information for the transaction.

Table B-1 IBMQREP_EXCEPTIONS table

Column name	**Data type**	**Description**
EXCEPTION_TIME	TIMESTAMP NOT NULL WITH DEFAULT	The timestamp at the Q Apply server when the error or conflict occurred.
RECVQ	VARCHAR(48) NOT NULL	The name of the receive queue where the transaction message arrived.
SRC_COMMIT_LSN	VARCHAR(48) FOR BIT DATA NOT NULL	The logical log sequence number at the Q Capture server for the transaction.
SRC_TRANS_TIME	TIMESTAMP NOT NULL	The timestamp at the Q Capture server for the transaction.
SUBNAME	VARCHAR(132) NOT NULL	The name of the Q subscription that the transaction belonged to.
REASON	CHAR(12) NOT NULL	A description of the error or conflict that caused the transaction to be logged into the IBMQREP_EXCEPTIONS table. “NOTFOUND” - An attempt to delete or update a row that did not exist. DUPLICATE - An attempt to insert a row that was already present. “CHECKFAILED” - The conflict detection rule was to check all values or check changed values, and a non-key value was not as expected. “SQLERROR” - An SQL error occurred, and it was not on the list of acceptable errors in the OKSQLSTATES column of the IBMQREP_TARGETS table. “OKSQLSTATE” - An SQL error occurred, and it was on the list of acceptable errors in the OKSQLSTATES column of the IBMQREP_TARGETS table. “P2PDUPKEY” - In peer-to-peer replication, a key update failed because a target row with the same key already existed, but was newer. “P2PNOTFOUND” - In peer-to-peer replication, a delete or update failed because the target row did not exist. “P2PVERLOSER” - In peer-to-peer replication, a delete or update failed because the target row was newer than the row in the change message.
SQLCODE	INTEGER	The SQL code returned by DB2 for the transaction.

[1] A schema is a file that defines what tags are legal in an XML document. For messages from the Q Capture program, the schema document is mqcap.xsd.

Column name	Data type	Description
SQLSTATE	CHAR(5)	The SQL state number returned by DB2 for the transaction.
SQLERRMC	CHAR(70) FOR BIT DATA	The error message tokens from the SQLCA structure used for executing the transaction.
OPERATION	VARCHAR(18) NOT NULL	The type of SQL operation that failed. Possible values are INSERT, INSERT(LOAD), DELETE, DELETE(LOAD), UPDATE, UPDATE(LOAD), KEY UPDATE, KEY UPDATE(LOAD).
TEXT	CLOB (32768) NOT NULL	An XML description of the data from the row that caused an error. The message is encoded using the same schema used for an XML publication. The root element is either insertRow, deleteRow, or updateRow. Both before and after values are included when available. The XML document is encoded in the target database codepage. For client applications that are using the IBMQREP_EXCEPTIONS table, the encoding XML header attribute of the document will indicate the target database codepage, but the document will be in the client application's codepage.
IS_APPLIED	CHAR(1) NOT NULL	A flag that indicates whether the row was applied to the target table even though it was entered into the IBMQREP_EXCEPTIONS table. Values are: ▶ "Y" - The row was applied because the CONFLICT_ACTION specified for the Q subscription was "F" (force). ▶ "N" - The transaction was not applied.
CONFLICT_RULE	CHAR(1)	The type of conflict detection that resulted in the row being entered in the IBMQREP_EXCEPTIONS table. Values are: ▶ "K" - Only key values were checked. ▶ "C" - Changed non-key values as well as key values were checked. ▶ "A" - All values were checked.
REPLROWID	ROWID NOT NULL GENERATED BY DEFAULT	Automatically generated when LOB columns exist.

Note: In addition to recording exceptions in the IBMQREP_EXCEPTIONS table, the Q Apply program also writes entries to the Q Apply log and the IBMQREP_APPLYTRACE table.

THE Q Apply log contains a diagnostic message in case of error or conflict, but does not include row information, which is only written to the IBMQREP_EXCEPTIONS table. The Q Apply log does not identify the actual conflict actions either.

The IBMQREP_APPLYTRACE table contains informational, warning, and error messages.

Additionally, the IBMQREP_APPLYMON table has a column "ROWS_NOT_APPLIED", which contains the number of rows that could not be applied, and were entered in the IBMQREP_EXCEPTIONS table.

When unexpected conditions occur, the user has a number of options available to specify the action to be taken by the Q Apply program.

For full details on exception processing, refer to the DB2 Information Center:

`http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp`

In the following sections, we describe the following:

- Conflict considerations
- SQL errors considerations
- Viewing exceptions in the IBMQREP_EXCEPTIONS table
- Conflict detection in bidirectional high availability scenarios

B.1.1 Conflict considerations

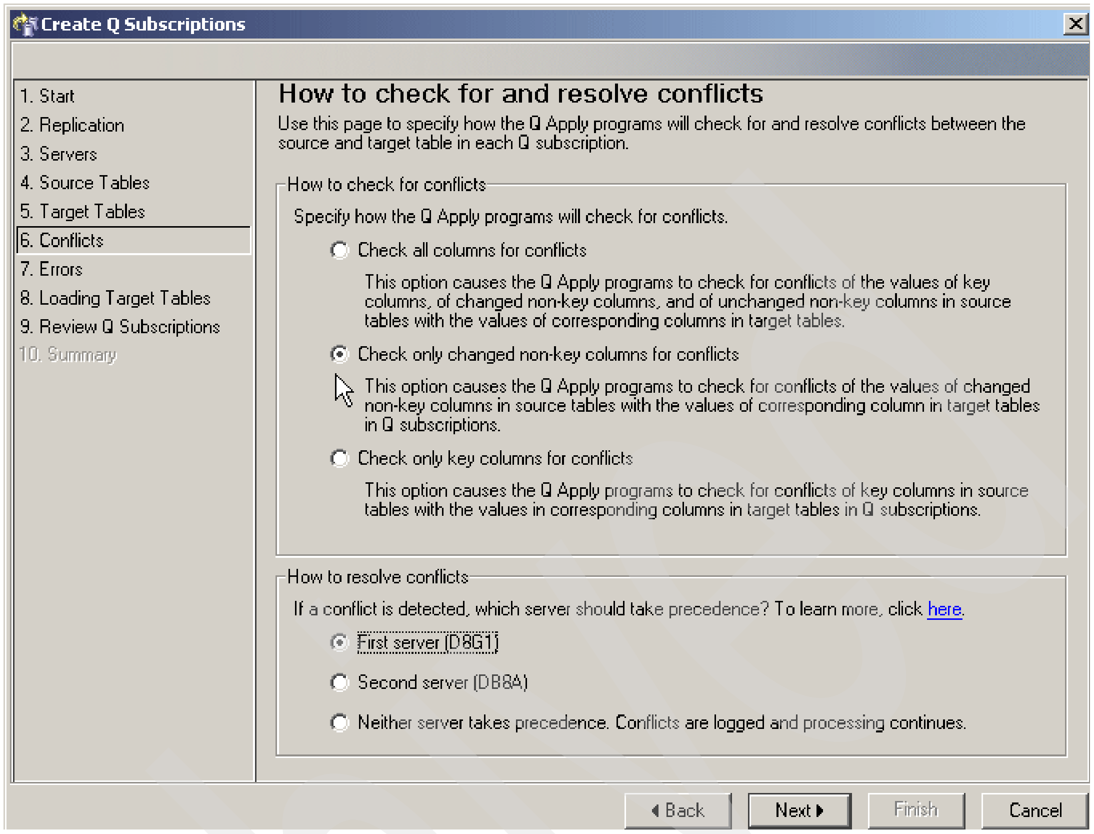
Conflicts can occur when an application other than the Q Apply program is updating or has updated the target table. For example, a conflict exists when the Q Apply program tries to insert a row that already exists.

The following subsections describe the conflict rules and the conflict action that you may specify for a given subscription.

B.1.1.1 Conflict rule

For a given subscription, you can specify what conditions the Q Apply program should check for to determine whether a conflict exists. These are called conflict rules.

For bidirectional replication, the conflict rule specifiable is as shown in Figure B-1.



How to check for and resolve conflicts

Use this page to specify how the Q Apply programs will check for and resolve conflicts between the source and target table in each Q subscription.

How to check for conflicts

Specify how the Q Apply programs will check for conflicts.

( ) Check all columns for conflicts

This option causes the Q Apply programs to check for conflicts of the values of key columns, of changed non-key columns, and of unchanged non-key columns in source tables with the values of corresponding columns in target tables.

(✓) Check only changed non-key columns for conflicts

This option causes the Q Apply programs to check for conflicts of the values of changed non-key columns in source tables with the values of corresponding column in target tables in Q subscriptions.

( ) Check only key columns for conflicts

This option causes the Q Apply programs to check for conflicts of key columns in source tables with the values in corresponding columns in target tables in Q subscriptions.

How to resolve conflicts

If a conflict is detected, which server should take precedence? To learn more, click here.

(✓) First server (D8G1)

( ) Second server (DB8A)

( ) Neither server takes precedence. Conflicts are logged and processing continues.

Figure B-1 Conflict rule for a subscription in bidirectional Q replication

The conflict rule choices are as follows:

- Check all columns for conflicts (CONFLICT_RULE ‘A’ in the IBMQREP_TARGETS table).

  With this option, the Q Apply program attempts to update or delete the target row by checking all columns that are in the target table.

  With this conflict rule, the Q Capture program sends the greatest amount of data to the Q Apply program for conflict checking.

- Check only changed non-key columns for conflicts (CONFLICT_RULE ‘C’ in the IBMQREP_TARGETS table)’.

  With this option, the Q Apply program attempts to update the target row by checking the key columns and the columns that changed in the update. The Q Apply program detects the following conflicts:

  – A row is not found in the target table.

  – A row is a duplicate of a row that already exists in the target table.

  – A row is updated at both servers simultaneously and the same column values changed.

  Note: If a row is updated at both servers simultaneously and different column values are changed, then there is no conflict.

  With this conflict rule, the Q Apply program merges updates that affect different columns in the same row. Because the Q Apply program requires the before values for changed columns for this conflict action, the Q Capture program sends the before values of changed columns.

Attention: With this option, the Q Apply program attempts to delete the target row by checking *all* columns that are in the target table even though the conflict rule is ‘C’.

- Check only key columns for conflicts (CONFLICT_RULE ‘K’ in the IBMQREP_TARGETS table).

  This is the default. With this option, the Q Apply program attempts to update or delete the target row by checking the values in the key columns. The Q Apply program detects the following conflicts:

  – A row is not found in the target table.

  – A row is a duplicate of a row that already exists in the target table.

  With this conflict rule, the Q Capture program sends the least amount of data to the Q Apply program for conflict checking. No before values are sent, and only the after values for any changed columns are sent.

Restrictions for LOB data types: Because before values are used to detect conflicts in bidirectional replication and Q replication does not replicate before values for LOB data types, conflicts in LOB columns are *not* detected.

Attention: For peer-to-peer replication, the only conflict rule supported is CONFLICT_RULE ‘V’ in the IBMQREP_TARGETS table, which indicates that the Q Apply program only checks the version column before applying a row.

Example B-1 summarizes the conflict rule options and their applicability to bidirectional and peer-to-peer replication environments.

Example: B-1 Conflict rule considerations

Conflict rule	**Conflict value in CONFLICT_RULE column of the IBMQREP_TARGETS table**	**Applies to**
Only key columns (default)	‘K’	Bidirectional replication only
Key columns and changed columns	‘C’	Bidirectional replication only
All columns	‘A’	Bidirectional replication only
Version columns only	‘V’	Peer-to-peer replication only

There are therefore two broad categories of conflict detection: value-based conflicts and version-based conflicts:

- Value-based conflicts

  These occur in a bidirectional Q replication environment, such as when an attempt is made to delete or update a row that did not exist at the target, or to insert a row that already exists. The Q Apply program records the type of conflict detection that was used (CONFLICT_RULE of ‘K’, ‘C’, or ‘A’) in the IBMQREP_EXCEPTIONS table.

- Version-based conflicts

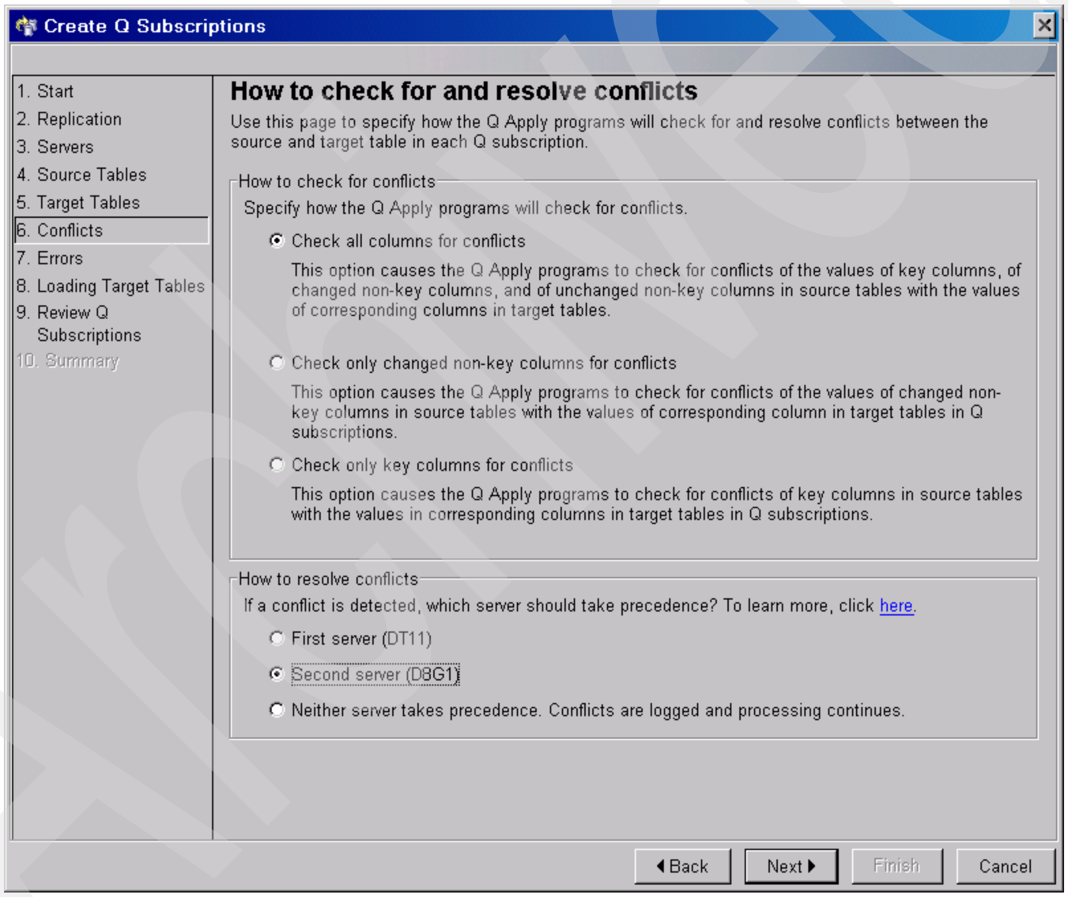
  These occur only in a peer-to-peer Q replication environment, such as when an attempt is made to update a row with the values from an older row, or an attempt is made to insert a row when a newer row with the same key already exists at the target.

In both the value-based conflicts and version-based conflicts cases, the Q Apply program also records the action taken when such a conflict occurs, as described in Section B.1.1.2, “Conflict action” on page 197.

B.1.1.2 Conflict action

For a given subscription, you can specify what action the Q Apply program should take when a conflict is detected.

The conflict actions available only apply to bidirectional replication and are specifiable as shown in Figure B-2.



How to check for and resolve conflicts

Use this page to specify how the Q Apply programs will check for and resolve conflicts between the source and target table in each Q subscription.

How to check for conflicts

Specify how the Q Apply programs will check for conflicts.

[✓ Check all columns for conflicts]

This option causes the Q Apply programs to check for conflicts of the values of key columns, of changed non-key columns, and of unchanged non-key columns in source tables with the values of corresponding columns in target tables.

Check only changed non-key columns for conflicts

This option causes the Q Apply programs to check for conflicts of the values of changed non-key columns in source tables with the values of corresponding column in target tables in Q subscriptions.

Check only key columns for conflicts

This option causes the Q Apply programs to check for conflicts of key columns in source tables with the values in corresponding columns in target tables in Q subscriptions.

How to resolve conflicts

If a conflict is detected, which server should take precedence? To learn more, click here.

First server (DT11)

[✓ Second server (D8G1)]

Neither server takes precedence. Conflicts are logged and processing continues.

Figure B-2 Conflict action for a subscription in bidirectional Q replication

For each server, you can choose what action each server takes when a conflict occurs. Each server can either force the conflicting row into its target table or ignore the conflict as follows:

- Ignore the unexpected condition (CONFLICT_ACTION ‘I’ in the IBMQREP_TARGETS table).

  This is the default. With this option, when the Q Apply program encounters an unexpected condition in the target data, the Q Apply program ignores the unexpected condition, does not apply the row, logs the condition and any rows that it did not apply, and then completes and commits the rest of the transaction. Whatever data is at the target wins—the target data is not overwritten. However, all rows that the are not applied are logged in the IBMQREP_EXCEPTIONS table.

- Force the change (CONFLICT_ACTION 'F' in the IBMQREP_TARGETS table).

  With this option, when the Q Apply program encounters an unexpected condition in the target data, the Q Apply program forces the change from the source table into the target table or the parameters for the stored procedure. The Q Apply program changes the operation (for example, from an insert to an update) if necessary, so that it can be applied to the target table or passed to the stored procedure parameters. Then it tries to reapply the change.

These options of force and ignore can be paired in two different ways to provide different behaviors for the Q Apply program.

1. One server forces conflicts, the other server ignores conflicts.

   One server (the one with the conflict action of ignore) wins if a conflict occurs. This server is the master or source owner of the data. If a row is updated at both servers and the same column values changed, then the master server (the one with the conflict action of ignore) ignores the conflict, and the row from the master server is forced in the target table on the other server (the one with the conflict action of force). For this conflict action, the Q Capture program sends the before values of all columns to the Q Apply program. The Q Apply program logs all conflicts in the IBMQREP_EXCEPTIONS table.

   Attention: For a high availability bidirectional replication environment, we recommend that the primary server be defined with the conflict action of force, and the secondary server be defined with the conflict action of ignore.

2. Both servers ignore conflicts.

   Any time a conflict occurs because a row is not found or a row is a duplicate of a row that already exists in the target table, the Q Apply program logs the conflict in the IBMQREP_EXCEPTIONS table, but otherwise ignores the conflict. This conflict action specifies that the Q Capture program does not send before values to the Q Apply program for conflict checking. Only the after values for any changed columns are sent.

   Note: Set both servers to ignore conflicts if you do not expect any conflicts to occur between the two servers and you want the least overhead to be used for conflict detection by the Q Capture and Q Apply programs. This is not applicable to a high availability bidirectional replication environment.

Attention: For peer-to-peer replication, there is no conflict action supported since Q Apply either ignores (when the timestamp is older) or forces (when the timestamp is newer) in the case of a conflict.

Example B-2 summarizes the conflict action options and their applicability to bidirectional and peer-to-peer replication environments.

Example: B-2 Conflict action considerations

Conflict rule	**Conflict value in CONFLICT_ACTION column of the IBMQREP_TARGETS table**	**Applies to**
Ignore (default)	‘I’	Bidirectional replication only
Force the change	‘F’[a]	Bidirectional replication only
Not specifiable		Peer-to-peer replication

a. Although CONFLICT_ACTION is not specifiable for peer-to-peer application, admin tools still put an ‘F’ in the IBMQREP_TARGETS table.

Very important: When the Q Apply program detects a conflict for a given row, it acts only on that row. In either bidirectional or peer-to-peer replication, the conflicting row is either accepted as a change to the target, or it is ignored and not applied. Because the goal of peer-to-peer replication is to provide a convergent set of copies of a database table, the conflicting row is acted upon, not the entire transaction. The practice of rejecting or accepting whole transactions is likely to quickly lead to a set of database copies that do not converge. The Q Apply program reports all conflicting rows in the IBMQREP_EXCEPTIONS table. In a peer-to-peer configuration, which might include several servers, Q replication attempts to report the conflicting row only once. But in some cases a conflict might show up in the IBMQREP_EXCEPTIONS table on more than one server. These duplications are easy to see because the data values and versioning information are identical. To see the complete conflict activity for a peer-to-peer or bidirectional configuration, look at the IBMQREP_EXCEPTIONS tables of all servers.

B.1.2 SQL errors considerations

SQL errors can occur for a broad range of reasons. One possible reason for an SQL error is if the Q Apply program tries to insert a child row at the target that is missing the corresponding parent row at the target.

SQL errors encountered by the Q Apply program fall into two categories: acceptable errors and unexpected errors.

- Acceptable SQL errors

  Rows that caused SQL errors that were defined as acceptable. When acceptable errors are encountered by the Q Apply program, the error action is similar to the ignore conflict rule discussed earlier. The SQL error is logged in the IBMQREP_EXCEPTIONS table, and the Q Apply program moves on to process the next row received from the source. Q Any changes made in that unit-of-work up to this point are not rolled back, and Q Apply commits the DB2 transaction when the last message in this transaction has been processed.

  You define acceptable SQL errors when you create a Q subscription via the Replication Center, as shown in Figure B-3, and using SQL, as shown in Example B-3.

  Note: SQL codes +100 and -803 are processed by Q Apply before it checks for OKSQLSTATEs, which means that the states corresponding to +100 and -803 are ignored by Q Apply.

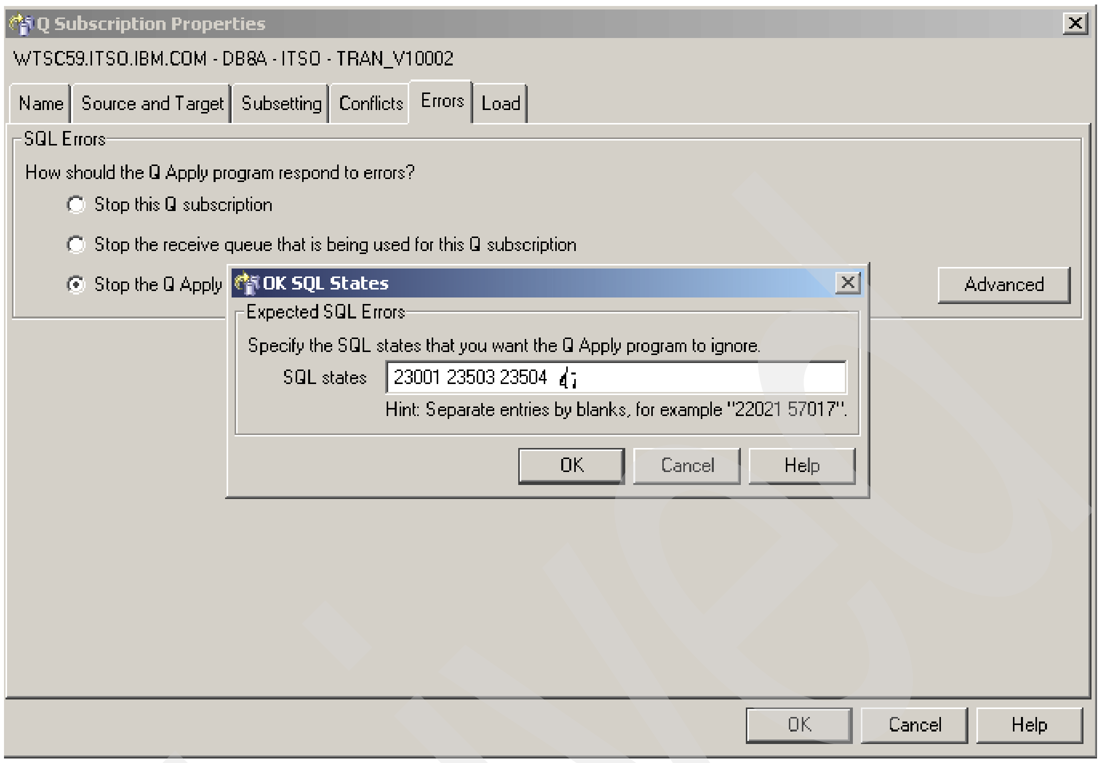

Q Subscription Properties
WTSC59.ITSO.IBM.COM - DB8A - ITSO - TRAN_V10002
Name
Source and Target
Subsetting
Conflicts
Errors
Load
SQL Errors
How should the Q Apply program respond to errors?
Stop this Q subscription
Stop the receive queue that is being used for this Q subscription
Stop the Q Apply
Advanced
OK SQL States
Expected SQL Errors
Specify the SQL states that you want the Q Apply program to ignore.
SQL states
23001 23503 23504
Hint: Separate entries by blanks, for example "22021 57017".
OK
Cancel
Help
OK
Cancel
Help

Figure B-3 Specifying OKSQLSTATES for a subscription in the Replication Center

Example: B-3 Specifying OKSQLSTATES for a subscription using SQL

```
update itso.ibmqrep_targets set oksqlstates = '23001 23503 23504 23505' where subname =
'TRAN_V10002'
```

- Unexpected SQL errors

  Rows that caused SQL errors that were not defined as acceptable for the Q subscription. When unexpected SQL errors are encountered by the Q Apply program, the action taken depends upon the error action specified. This is described Section B.1.2.1, "Error action" on page 200.

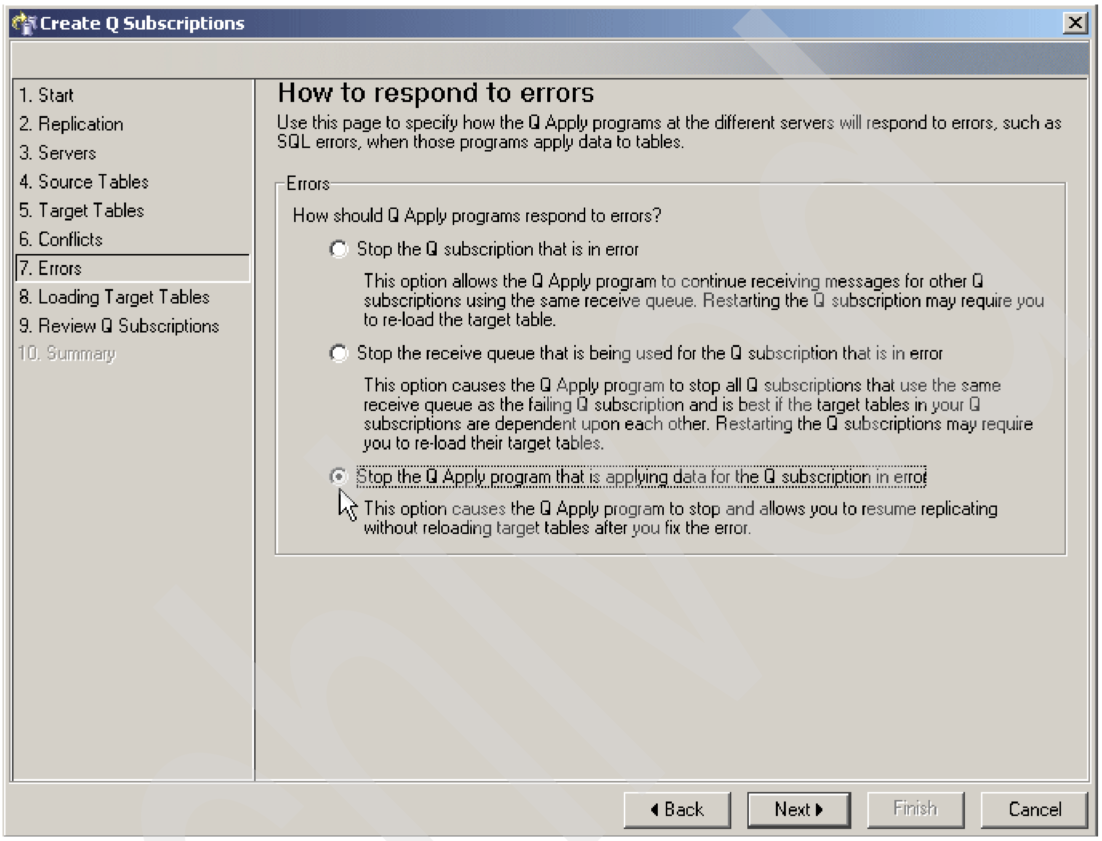
Attention: Regardless of whether the errors encountered by the Q Apply program are acceptable SQL errors or unexpected SQL errors, it saves the SQL code and SQL state code returned by DB2 for the transaction, as well as the error message tokens from the SQLCA structure that is used for executing the transaction.

B.1.2.1 Error action

In Q replication, you can specify what action the Q Apply program takes when it encounters errors, such as SQL errors, in your environment. The option that you choose depends on the level of granularity at which you want to isolate and fix the problem. The same error options apply for unidirectional, bidirectional, and peer-to-peer replication.

You can choose for one of the following actions to occur when the Q Apply program encounters *acceptable* SQL errors, as shown in Figure B-4. This applies to both bidirectional and peer-to-peer replication environments.

Important: Error action only applies to updates performed by user transactions, and not to other errors encountered by the Q Apply program such as being unable to connect to the primary server during an automatic load.



How to respond to errors

Use this page to specify how the Q Apply programs at the different servers will respond to errors, such as SQL errors, when those programs apply data to tables.

Errors

How should Q Apply programs respond to errors?

( ) Stop the Q subscription that is in error

This option allows the Q Apply program to continue receiving messages for other Q subscriptions using the same receive queue. Restarting the Q subscription may require you to re-load the target table.

( ) Stop the receive queue that is being used for the Q subscription that is in error

This option causes the Q Apply program to stop all Q subscriptions that use the same receive queue as the failing Q subscription and is best if the target tables in your Q subscriptions are dependent upon each other. Restarting the Q subscriptions may require you to re-load their target tables.

[✓ Stop the Q Apply program that is applying data for the Q subscription in error]

This option causes the Q Apply program to stop and allows you to resume replicating without reloading target tables after you fix the error.

Figure B-4 Error action for a subscription - both bidirectional and peer-to-peer replication

The error action choices are as follows:

- Stop the Q subscription that is in error.

  Deactivate the corresponding Q subscription (ERROR_ACTION 'D' in the IBMQREP_TARGETS table) as described earlier for the conflict action in Section B.1.1.2, "Conflict action" on page 197.

- Stop the receive queue that is being used for the Q subscription that is in error.

  Have the Q Apply program stop reading from the corresponding receive queue (ERROR_ACTION 'Q' in the IBMQREP_TARGETS table) as described earlier for the conflict action in Section B.1.1.2, "Conflict action" on page 197.

  This is the default.

- Stop the Q Apply program that is applying data for the Q subscription in error.

  Stop the Q Apply program (ERROR_ACTION 'S' in the IBMQREP_TARGETS table) as described earlier for the conflict action in Section B.1.1.2, "Conflict action" on page 197.

Example B-4 summarizes the error action options and their applicability to bidirectional and peer-to-peer replication environments.

Example: B-4 Error action considerations

Error action	Error value in ERROR_ACTION column of the IBMQREP_TARGETS table	Applies to
Have the Q Apply program stop reading from the corresponding receive queue (default)	'Q'	Bidirectional and peer-to-peer replication
Deactivate the corresponding Q subscription	'D'	Bidirectional and peer-to-peer replication
Stop the Q Apply program	'S'	Bidirectional and peer-to-peer replication

B.1.3 Viewing exceptions in the IBMQREP_EXCEPTIONS table

The exceptions recorded in the IBMQREP_EXCEPTIONS table may be viewed using different facilities such as the Exceptions Table Formatter using command-line input, Replication Center, Live Monitor (which invokes the Exceptions Table Formatter), and SQL SELECT statements. Examples of each of these options follow.

B.1.3.1 Exceptions Table Formatter using command-line input

This tool accesses the contents of the IBMQREP_EXCEPTIONS table and formats its contents. The tool can be downloaded from:

http://www-1.ibm.com/support/docview.wss?&uid=swg27007070

Example B-5 shows an example of invoking this tool from the command line.

Example B-5 shows the output of the Exception Table Formatter for this exception. It describes the "Row Operation" being and 'Insert', which has an exception with a "REASON" of 'Duplicate', "SQL CODE" of '-803', and SQLSTATE" of '23505'. It identifies the "CONFLICT_RULE" as being C, and the conflict action "IS_APPLIED" of 'N', which corresponds to an ignore. The values in the discarded row are shown in the "Columns" portion of the output.

What the report does not show is the following:

- The values of the non-key columns that were detected in the existing row in the table.
- The individual columns whose values did not match—information that may be valuable if the "REASON" was identified as being 'CHECKFAILED'.

Example: B-5 Using Exceptions Table Formatter to view exceptions

```
The command to invoke the PrintExcpetionTable tool has the format:
PrintExceptionTable -db <db2 alias for qapply> -schema <qapply schema> -uid <db2 userid>
-pwd <db2 password>

C:\QExcFormat>PrintExceptionTable -db DB8A -schema ITSO -uid hueykim -pwd xxxxxxx


Exception Time     = 2006-02-28 14:17:24.111371
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE6EFF1F85330001'
  SRC_TRANS_TIME     = 2006-02-28 19:16:13.520307
  SUBNAME            = BAL_V10002
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
```

```
SQLERRMC  (EBCDIC) = BALRV1RP?0000000904
           (ASCII) = ????†¤??ˉ???????–?“
             (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F0F9F0F4
IS_APPLIED         = N
CONFLICT_RULE      = C
Database name      = DB8A
Row Operation      = Insert
Columns :
   Column Name = ACCT_ID
   Is Key      = true
   Value       = 4

   Column Name = ACCTG_RULE_CD
   Value       = A

   Column Name = BAL_TYPE_CD
   Value       = IN

   Column Name = BAL_DATE
   Value       = 2006-02-28

   Column Name = BAL_AMOUNT
   Value       = 100.0
```

B.1.3.2 Replication Center

Exception generated by the Q Apply program can be viewed via the Replication Center, as shown in Figure B-5 through Figure B-9 on page 207.

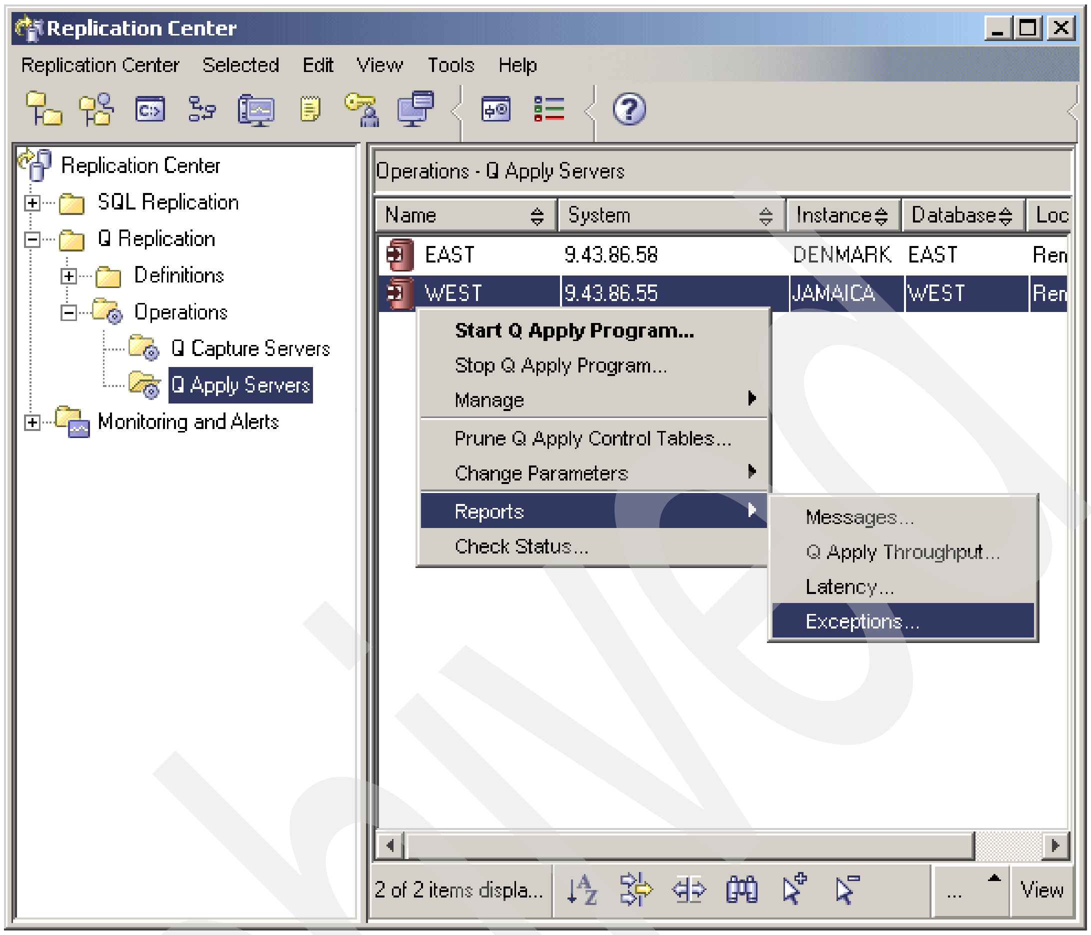
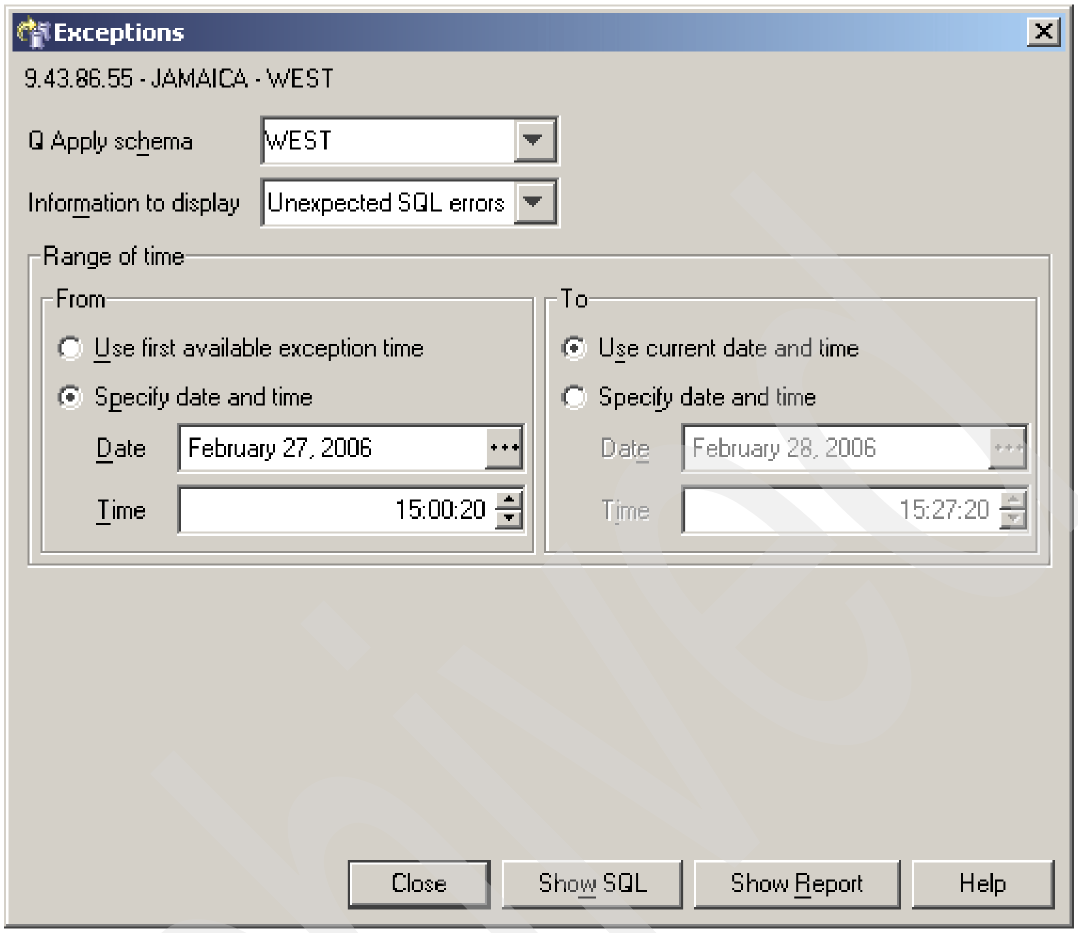


Figure B-5 Using Replication Center to view exceptions 1/5


Exceptions

9.43.86.55 - JAMAICA - WEST

Q Apply schema: WEST

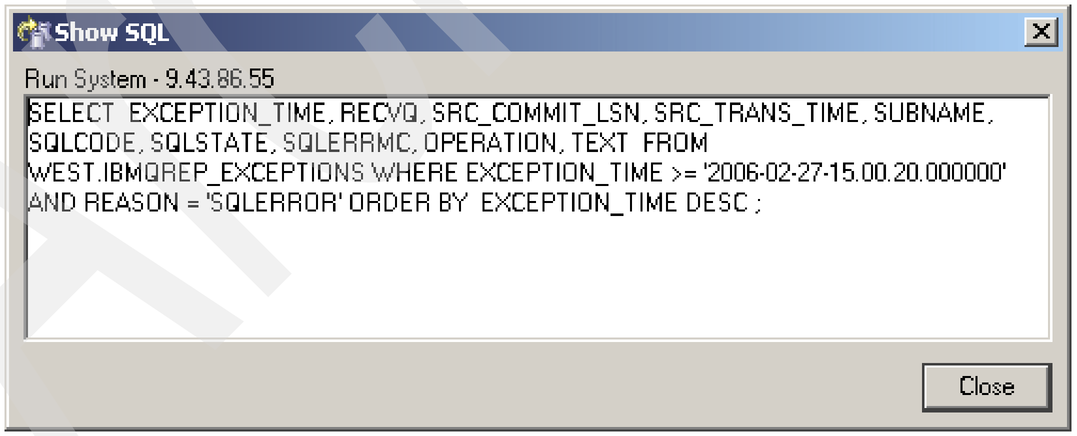
Information to display: Unexpected SQL errors

Range of time

From: ( ) Use first available exception time [✓ Specify date and time]
Date: February 27, 2006
Time: 15:00:20

To: [✓ Use current date and time] ( ) Specify date and time
Date: February 28, 2006
Time: 15:27:20

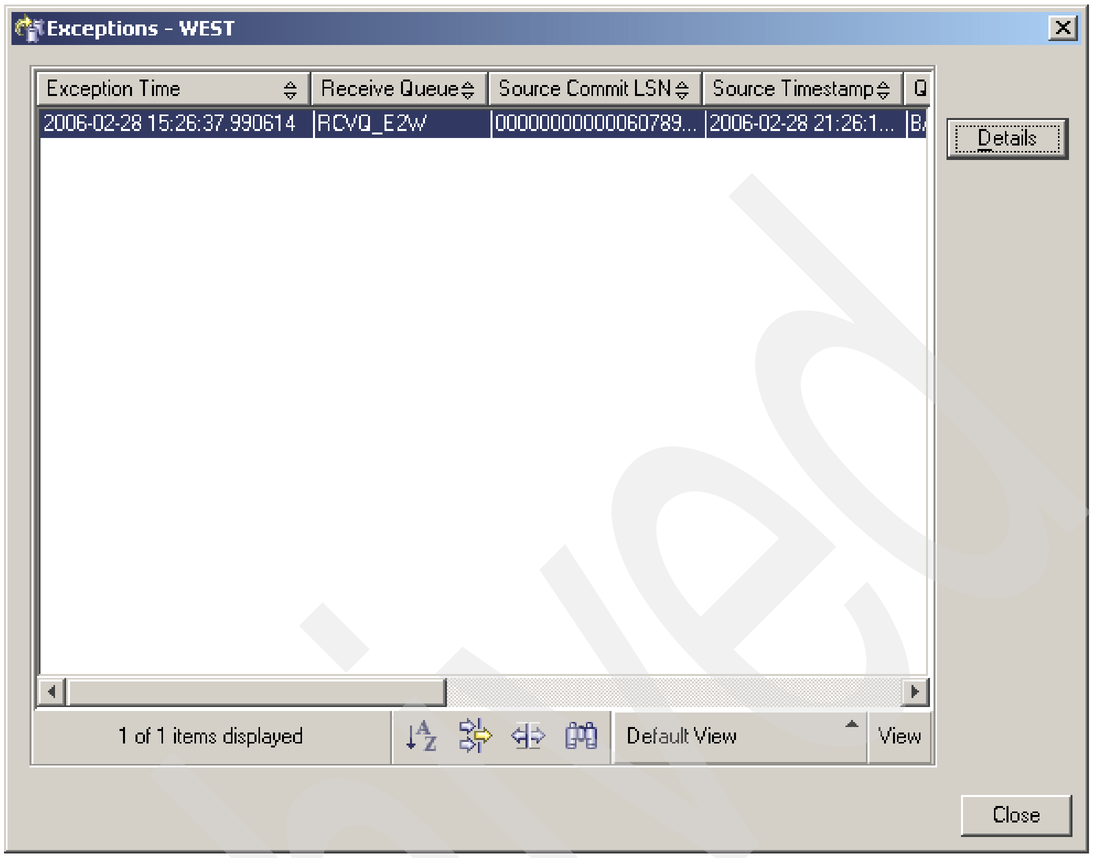
Figure B-6   Using Replication Center to view exceptions 2/5


Show SQL

Run System - 9.43.86.55

```
SELECT  EXCEPTION_TIME, RECVQ, SRC_COMMIT_LSN, SRC_TRANS_TIME, SUBNAME,
SQLCODE, SQLSTATE, SQLERRMC, OPERATION, TEXT  FROM
WEST.IBMQREP_EXCEPTIONS WHERE EXCEPTION_TIME >= '2006-02-27-15.00.20.000000'
AND REASON = 'SQLERROR' ORDER BY  EXCEPTION_TIME DESC ;
```

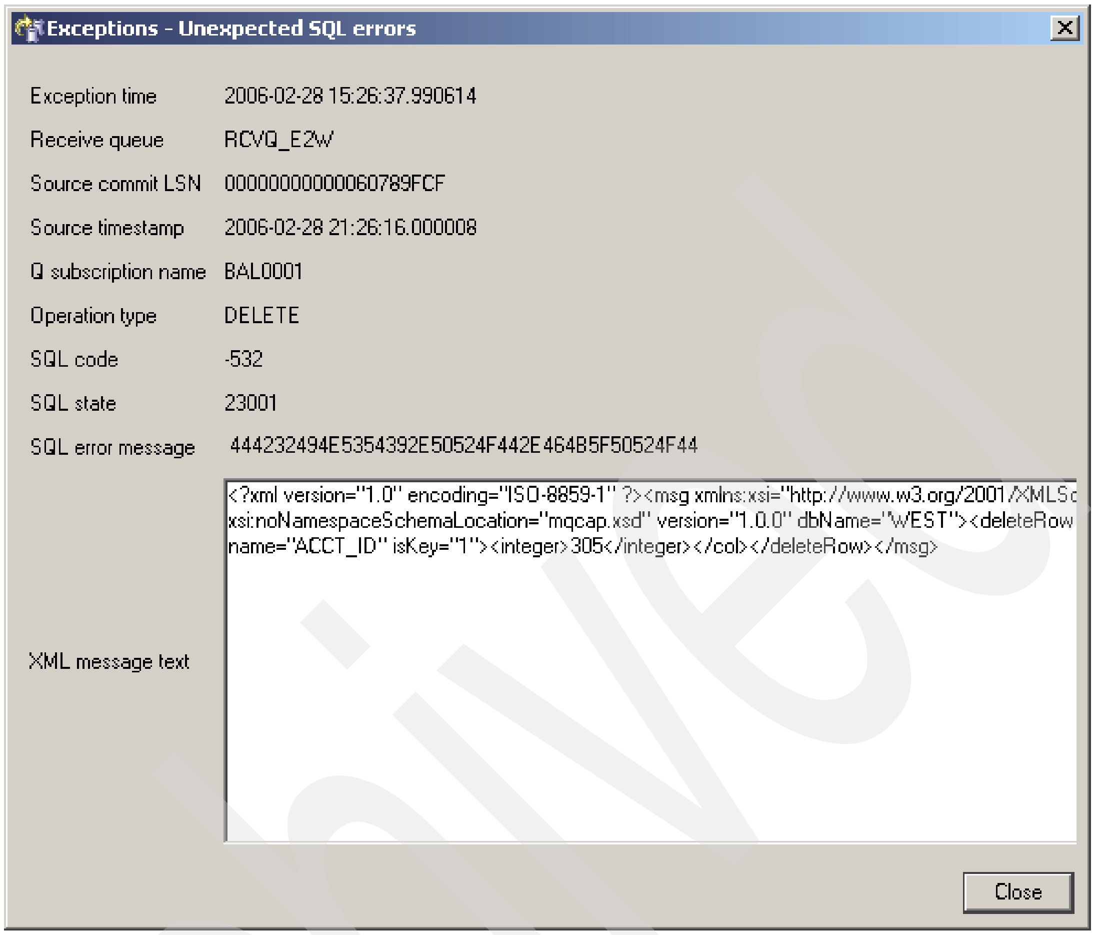
Figure B-7   Using Replication Center to view exceptions 3/5



Exception Time	Receive Queue	Source Commit LSN	Source Timestamp	Q
2006-02-28 15:26:37.990614	RCVQ_E2W	00000000000060789...	2006-02-28 21:26:1...	B/

1 of 1 items displayed

Figure B-8 Using Replication Center to view exceptions 4/5



Exception time	2006-02-28 15:26:37.990614
Receive queue	RCVQ_E2W
Source commit LSN	00000000000060789FCF
Source timestamp	2006-02-28 21:26:16.000008
Q subscription name	BAL0001
Operation type	DELETE
SQL code	-532
SQL state	23001
SQL error message	444232494E5354392E50524F442E464B5F50524F44
XML message text	<?xml version="1.0" encoding="ISO-8859-1" ?><msg xmlns:xsi="http://www.w3.org/2001/XMLSc xsi:noNamespaceSchemaLocation="mqcap.xsd" version="1.0.0" dbName="WEST"><deleteRow name="ACCT_ID" isKey="1"><integer>305</integer></col></deleteRow></msg>

Figure B-9 Using Replication Center to view exceptions 5/5

B.1.3.3 Live Monitor

This tool accesses the contents of the IBMQREP_EXCEPTIONS, IBMQREP_CAPMON, and IBMQREP_APPLYMON and other tables and graphically formats its contents. The tool can be also downloaded from:

http://www-1.ibm.com/support/docview.wss?&uid=swg270

Figure B-10 and Figure B-11 show how the contents of the exceptions table may be viewed via the Live Monitor tool.

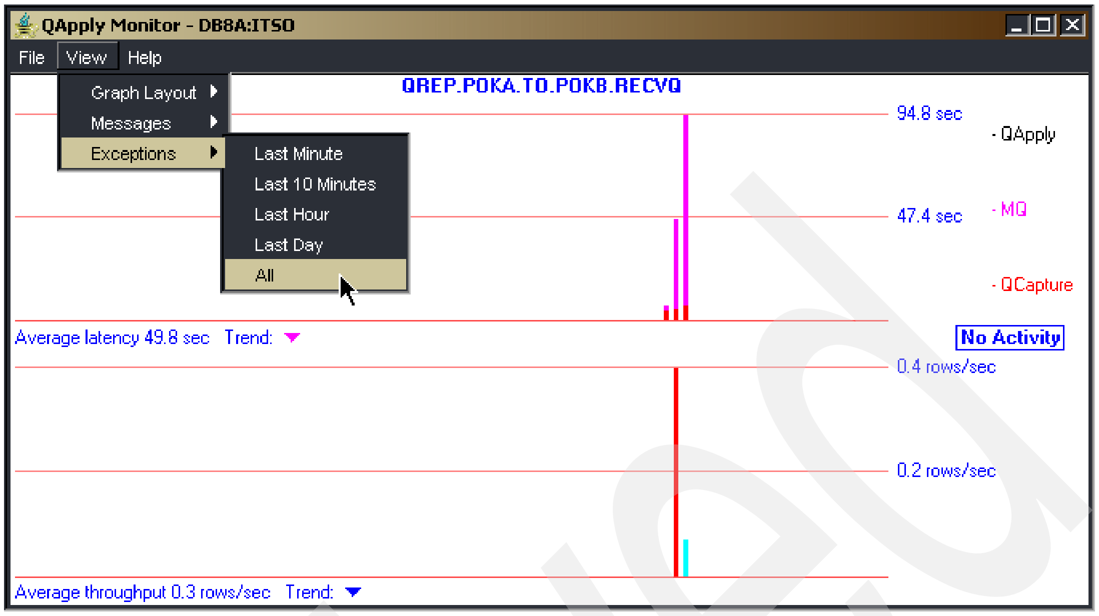
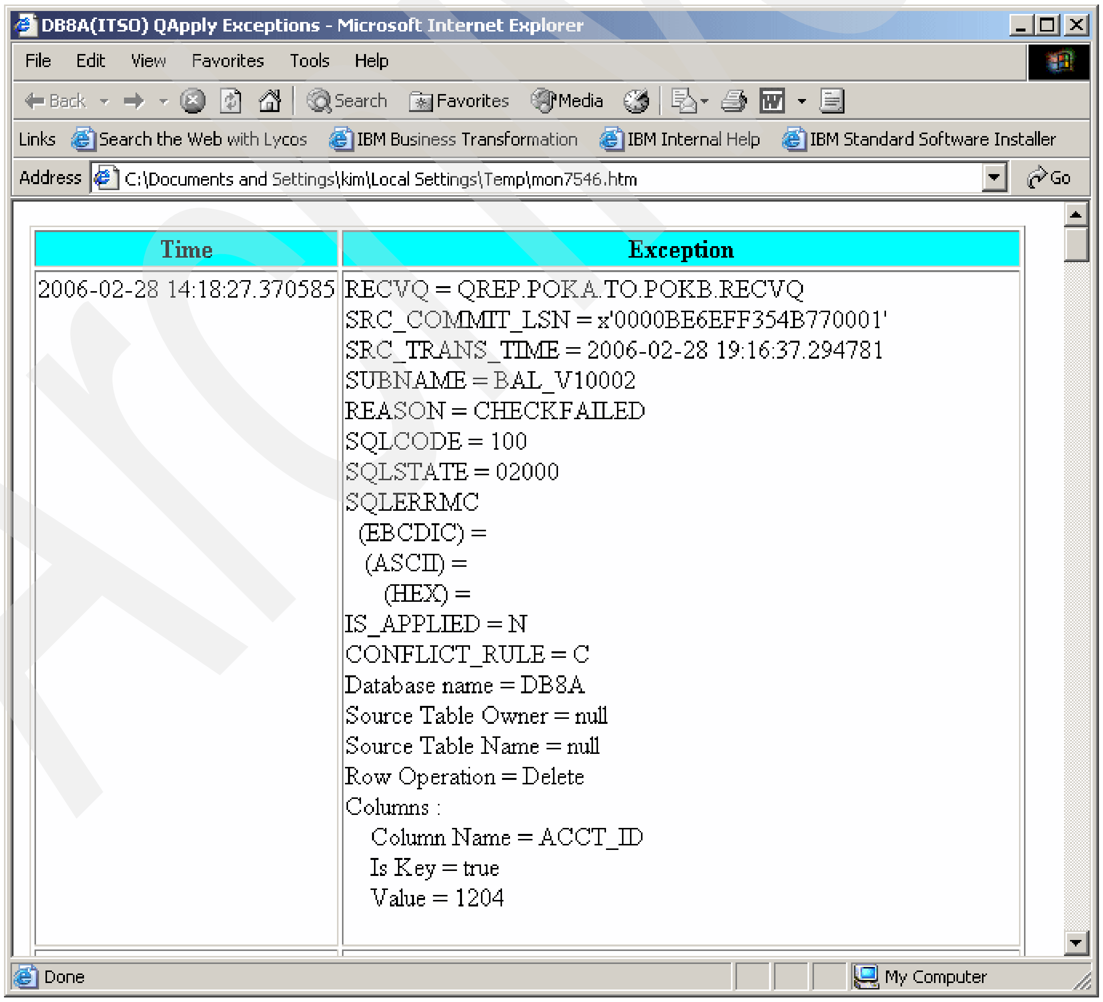


Figure B-10   Using Live Monitor to view exceptions 1/2



DB8A(ITSO) QApply Exceptions - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Links
Search the Web with Lycos
IBM Business Transformation
IBM Internal Help
IBM Standard Software Installer
Address
C:\Documents and Settings\kim\Local Settings\Temp\mon7546.htm
Go
Done
My Computer

Time	Exception
2006-02-28 14:18:27.370585	RECVQ = QREP.POKA.TO.POKB.RECVQ SRC_COMMIT_LSN = x'0000BE6EFF354B770001' SRC_TRANS_TIME = 2006-02-28 19:16:37.294781 SUBNAME = BAL_V10002 REASON = CHECKFAILED SQLCODE = 100 SQLSTATE = 02000 SQLERRMC (EBCDIC) = (ASCII) = (HEX) = IS_APPLIED = N CONFLICT_RULE = C Database name = DB8A Source Table Owner = null Source Table Name = null Row Operation = Delete Columns : Column Name = ACCT_ID Is Key = true Value = 1204

Figure B-11   Using Live Monitor to view exceptions 2/2

B.1.3.4 SQL SELECT statements

Example B-6 shows a simple SQL statement for viewing the contents of the IBMQREP_EXCEPTIONS table.

Example: B-6 Using SQL SELECT statements to view exceptions

```
SELECT * FROM EAST.IBMQREP_EXCEPTIONS ORDER BY EXCEPTION_TIME DESC
```

B.1.4 Considerations for conflict detection in high availability scenarios

As described in Section B.1.1, "Conflict considerations" on page 194, conflict detection in a bidirectional replication environment uses value-based conflict checking and involves choosing between the conflict rules of only key columns, key columns and changed columns, and all columns. Peer-to-peer replication, on the other hand, uses version-based checking that checks conflicts using the version columns.

Even the highest level of value-based conflict detection (checking all columns) in bidirectional replication is not as robust as version-based conflict detection in peer-to-peer replication. Some situations might result in a conflict not being detected, as described in Section B.3, "Data inconsistencies in an HA bidirectional scenario" on page 231.

Therefore, if you expect conflicts by application design, then choose a peer-to-peer configuration for robust HA environments. When application developers design an application that involves distributed copies of tables that can be updated at any server, the developers must fully explore the potential for conflicts, the impact of conflicts, and how conflicts will be resolved. Because conflicts can result in loss of durability of a transaction, applications should be designed to minimize the potential for conflicts.

B.2 Sample exceptions in a bidirectional scenario

In this section we describe common bidirectional Q replication scenarios that generate exceptions in the IBMQREP_EXCEPTIONS table, view these generated exceptions using the Exceptions Table Formatter, and analyze them.

The bidirectional Q replication HA environment used in the scenarios is shown in Figure B-12. The Active/Active read-only model during normal operations has the primary server (source) supporting the read/write workload, while the secondary server (target) supports a read-only workload. When the primary server goes down and becomes unavailable, the secondary server takes over the read/write workload. This is the failover process. Subsequently, when the primary server becomes available again, the read/write workload is transferred back to the primary server and the secondary server again only supports a read-only workload. This process of re-establishing normal operations from a failover environment is called switchback.

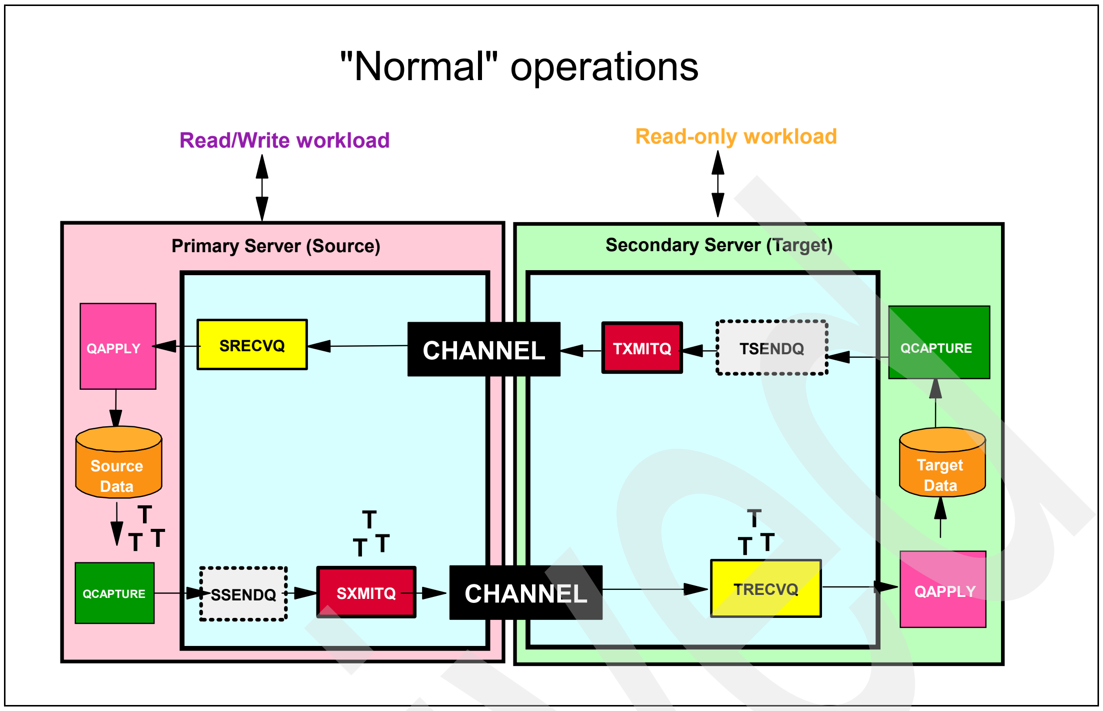


Figure B-12   Bidirectional Q replication HA environment - normal operations

When a failover to the secondary server (target) is triggered by a sudden failure of the primary server (source), there will most likely be data that was committed in the primary server (source) that did not get replicated over to the secondary server (target), as shown in Figure B-13.

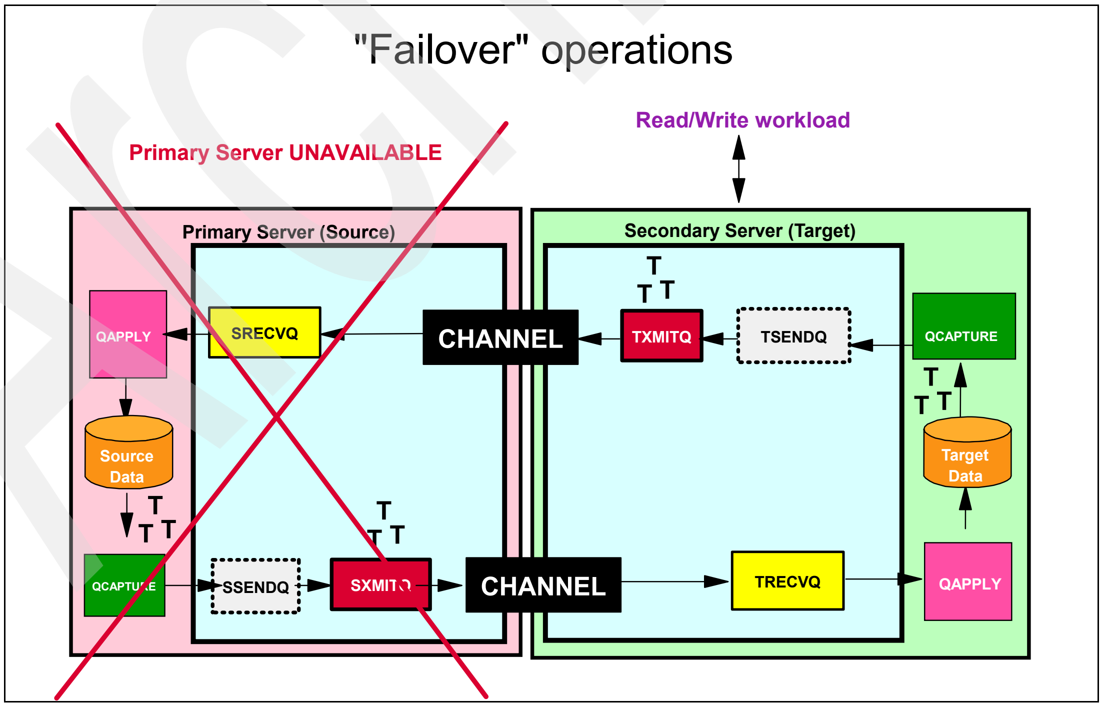


Figure B-13   Bidirectional Q replication HA environment - failover operations

This “residual” data (shown in Figure B-13 as a bunch of “T’s) may exist in the DB2 logs that had as yet not been captured by Q Capture, or in the transmit queue that had not yet made it over to the receive queue on the secondary server. Similarly, a potentially significant amount of committed data at the secondary server (target) after failover has not been propagated back to the primary server (source).

During switchback re-synchronization, as shown in Figure B-14, there is a significant potential for the unpropagated data (shown as a bunch of “T’s) to cause conflicts with the data on disk that result in exceptions being written to the IBMQREP_EXCEPTIONS tables at the primary server and secondary server. The conflict rule recommended for the HA environment is key columns and changed columns (option ‘C’) or all columns (option ‘A’). The conflict action at the primary server and target server depends upon the conflict action chosen for each subscription. For a HA environment, the recommended conflict action is ignore (option ‘I’) at the secondary server and force (option ‘F’) at the primary server.

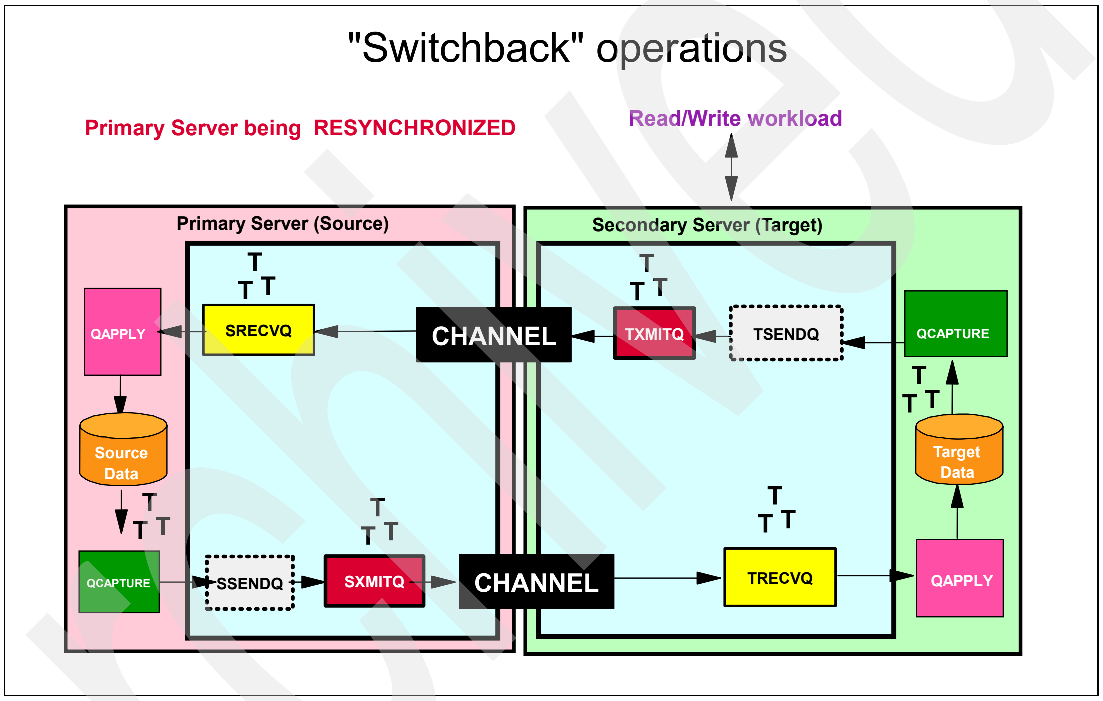


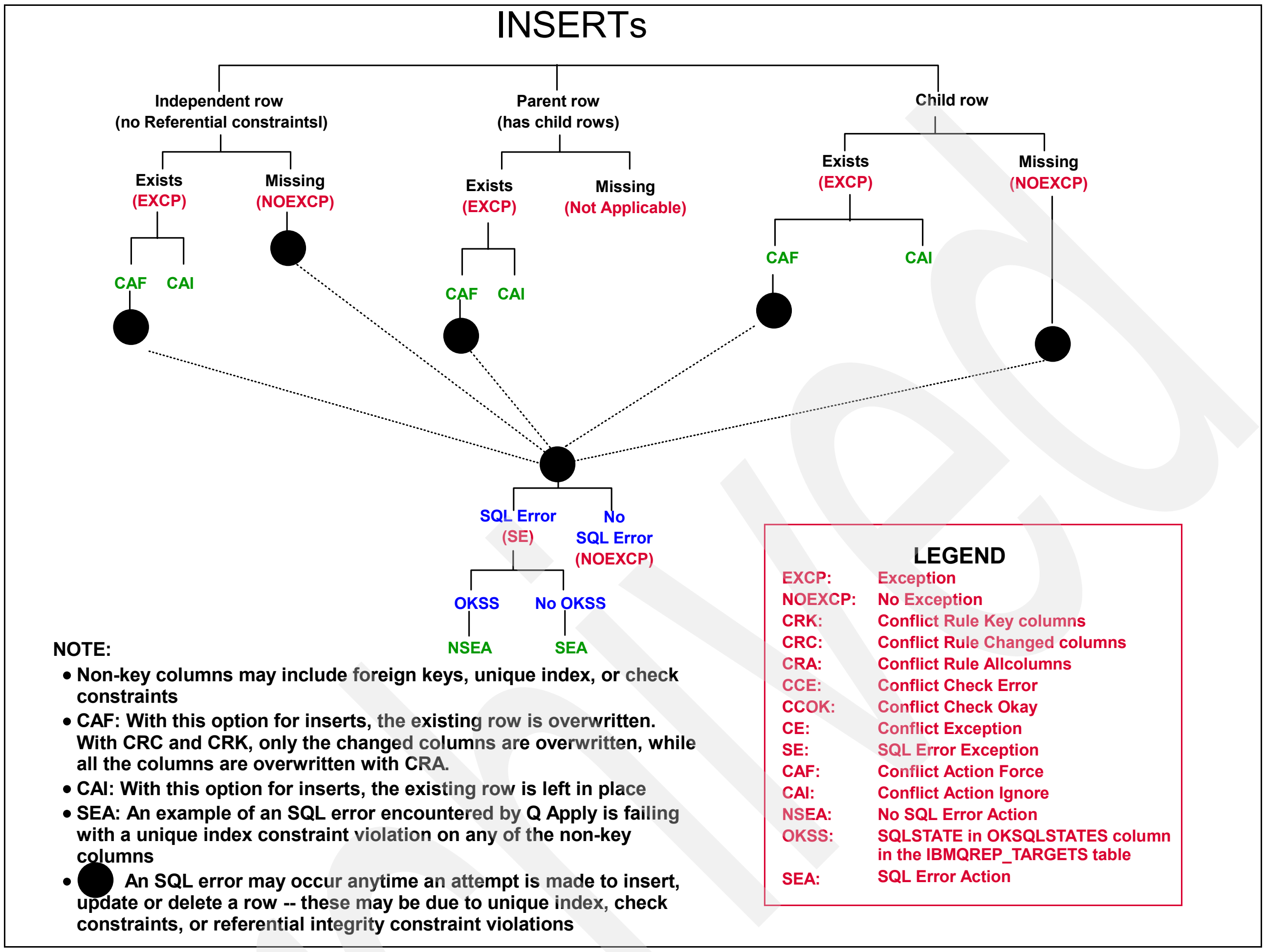
Figure B-14 Bidirectional Q replication HA environment - switchback operations

Figure B-15 through Figure B-17 describe the processing that occurs when inserts, updates, and deletes are propagated from the source to the target. These propagated changes may be accepted, rejected, or ignored at the target. In cases where the change is rejected or ignored at the target, an exception is recorded in the IBMQREP_EXCEPTIONS table at the target. These exceptions may be conflict exceptions or SQL error exceptions, and provide details of the action taken on the exception.

In Figure B-15 through Figure B-17 on page 214, the following definitions apply:

- An independent row is a row in a table that has no referential constraints defined. Therefore every row in the table is neither a parent row nor a child row. It is possible for a row in a table that has referential constrains defined to be an independent row if it does not participate in any referential relationship, that is, it is neither a parent row nor a child row.
- A parent row is a row in a table that has referential constraints defined, and the row has child rows associated with it.

- A child row is a row in a table that has referential constraints defined, and the row has one or more[2] parent rows associated with it.



NOTE:

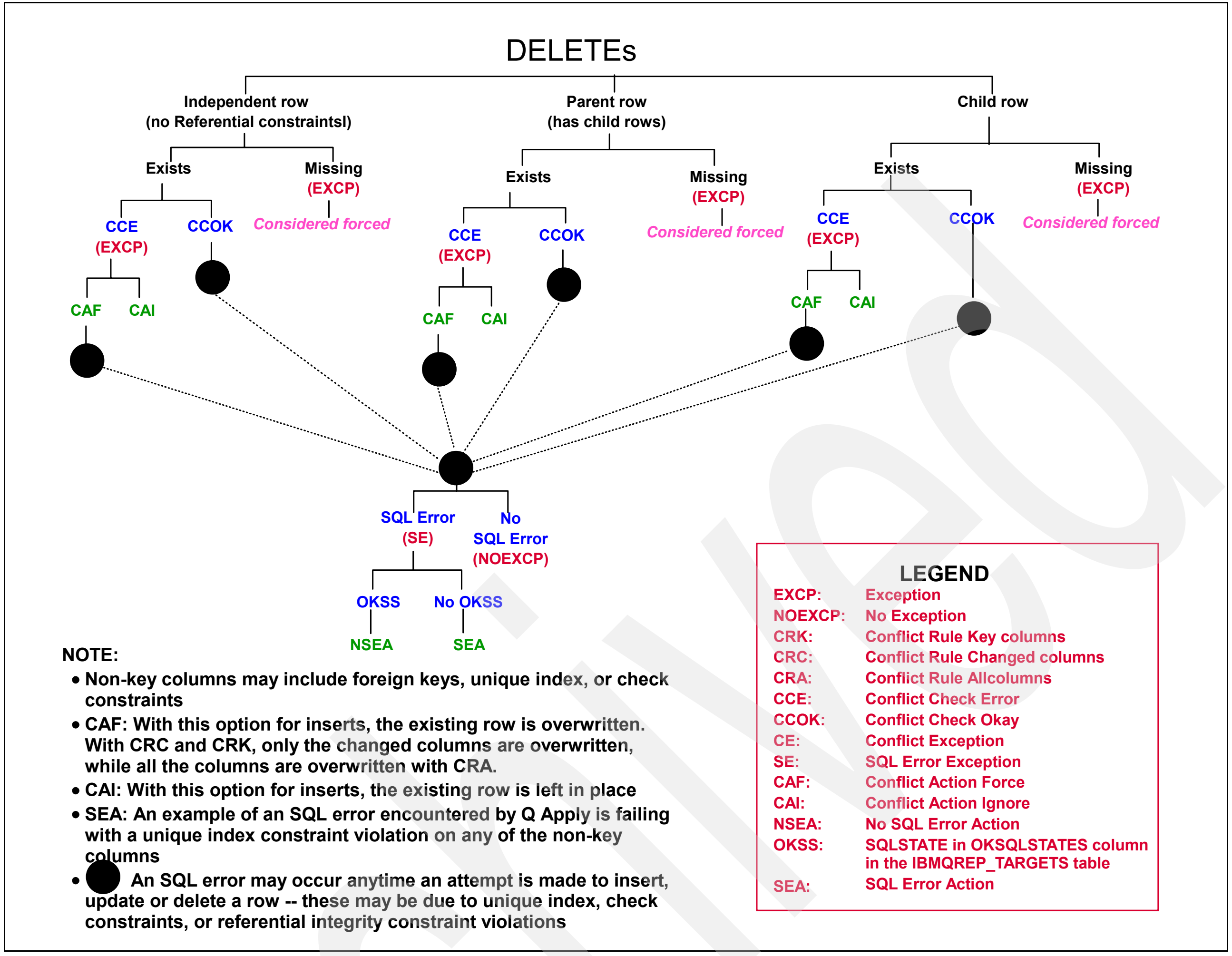
- Non-key columns may include foreign keys, unique index, or check constraints
- CAF: With this option for inserts, the existing row is overwritten. With CRC and CRK, only the changed columns are overwritten, while all the columns are overwritten with CRA.
- CAI: With this option for inserts, the existing row is left in place
- SEA: An example of an SQL error encountered by Q Apply is failing with a unique index constraint violation on any of the non-key columns
- ● An SQL error may occur anytime an attempt is made to insert, update or delete a row -- these may be due to unique index, check constraints, or referential integrity constraint violations

LEGEND

EXCP:	Exception
NOEXCP:	No Exception
CRK:	Conflict Rule Key columns
CRC:	Conflict Rule Changed columns
CRA:	Conflict Rule Allcolumns
CCE:	Conflict Check Error
CCOK:	Conflict Check Okay
CE:	Conflict Exception
SE:	SQL Error Exception
CAF:	Conflict Action Force
CAI:	Conflict Action Ignore
NSEA:	No SQL Error Action
OKSS:	SQLSTATE in OKSQLSTATES column in the IBMQREP_TARGETS table
SEA:	SQL Error Action

Figure B-15 Propagated INSERTs processing

[2] When a row has multiple non-null foreign keys associated with it



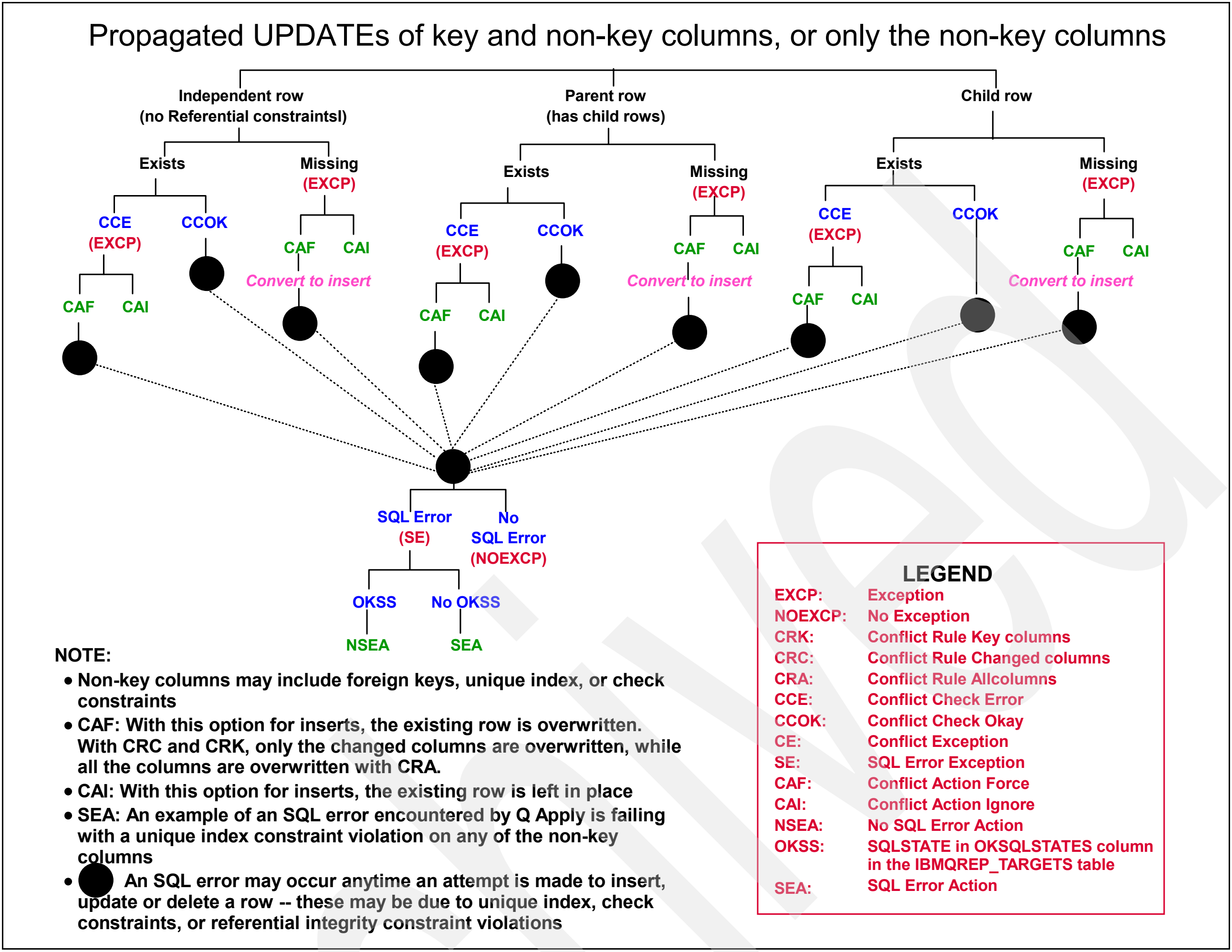
NOTE:

- Non-key columns may include foreign keys, unique index, or check constraints
- CAF: With this option for inserts, the existing row is overwritten. With CRC and CRK, only the changed columns are overwritten, while all the columns are overwritten with CRA.
- CAI: With this option for inserts, the existing row is left in place
- SEA: An example of an SQL error encountered by Q Apply is failing with a unique index constraint violation on any of the non-key columns
- ● An SQL error may occur anytime an attempt is made to insert, update or delete a row -- these may be due to unique index, check constraints, or referential integrity constraint violations

LEGEND

EXCP:	Exception
NOEXCP:	No Exception
CRK:	Conflict Rule Key columns
CRC:	Conflict Rule Changed columns
CRA:	Conflict Rule Allcolumns
CCE:	Conflict Check Error
CCOK:	Conflict Check Okay
CE:	Conflict Exception
SE:	SQL Error Exception
CAF:	Conflict Action Force
CAI:	Conflict Action Ignore
NSEA:	No SQL Error Action
OKSS:	SQLSTATE in OKSQLSTATES column in the IBMQREP_TARGETS table
SEA:	SQL Error Action

Figure B-16 Propagated DELETEs processing



NOTE:

- Non-key columns may include foreign keys, unique index, or check constraints
- CAF: With this option for inserts, the existing row is overwritten. With CRC and CRK, only the changed columns are overwritten, while all the columns are overwritten with CRA.
- CAI: With this option for inserts, the existing row is left in place
- SEA: An example of an SQL error encountered by Q Apply is failing with a unique index constraint violation on any of the non-key columns
- ● An SQL error may occur anytime an attempt is made to insert, update or delete a row -- these may be due to unique index, check constraints, or referential integrity constraint violations

LEGEND

EXCP:	Exception
NOEXCP:	No Exception
CRK:	Conflict Rule Key columns
CRC:	Conflict Rule Changed columns
CRA:	Conflict Rule Allcolumns
CCE:	Conflict Check Error
CCOK:	Conflict Check Okay
CE:	Conflict Exception
SE:	SQL Error Exception
CAF:	Conflict Action Force
CAI:	Conflict Action Ignore
NSEA:	No SQL Error Action
OKSS:	SQLSTATE in OKSQLSTATES column in the IBMQREP_TARGETS table
SEA:	SQL Error Action

Figure B-17 Propagated UPDATEs (key & non-key columns, or only the non-key columns) processing

Attention: In the following sample exception scenarios, we chose the following options:

- Conflict rule of key columns and changed columns (option ‘C’) for all subscriptions
- Conflict action of ignore (option ‘I’) at the secondary server for each subscription
- Conflict action of force (option ‘F’) at the primary server for each subscription
- Error action of stopping the Q Apply program that is applying data for the Q subscription in error (option ‘S’) at both the primary and secondary servers

The exceptions described here are categorized as follows:

- Exceptions with inserts
- Exceptions with deletes
- Exceptions with updates

B.2.1 Exceptions with inserts

In this section we describe exceptions generated by an insert of a row propagated from one server to another.

B.2.1.1 Insert independent/parent and the same key exists

This scenario corresponds to the case where a propagated row that is an independent/parent row encounters an existing row with the same key. The column values in the non-key columns of the propagated row may or may not match the corresponding non-key column values in the existing row. An exception is generated, and the propagated row may either be ignored (when the conflict action is ignore) or overwrite the existing row when the conflict action is force. Any conflict between any non-key values in the propagated row and that of the existing row are not identified.

Conflict action of force

Example B-7 shows the output of the Exception Table Formatter for this exception when the conflict action is force. It describes the “Row Operation” being an ‘Insert,’ which has an exception with a “REASON” of ‘Duplicate’, “SQL CODE” of ‘-803,’ and SQLSTATE” of ‘23505’. It identifies the “CONFLICT_RULE” as being ‘C’, and the conflict action “IS_APPLIED” of ‘Y,’ which corresponds to a force. The values in the overriding row are shown in the Columns portion of the output. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field). Also, no information is provided whether or not there were mismatches in the non-key columns between the propagated row and the existing row.

Note: In the case where the existing row is a parent row, the action performed is the same as if the existing row were an independent row, regardless of whether the delete rule is CASCADE or RESTRICT for the dependent tables.

Example: B-7 Insert independent/parent row and the same key exists - FORCE

```
Exception Time     = 2006-02-16 17:27:39.123015
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000007C4DD090000'
  SRC_TRANS_TIME     = 2006-02-16 22:24:54.300215
  SUBNAME            = BAL_V10001
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = BALRV1RP?000000021E
             (ASCII) = ????†¤??¯???????o¤
               (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F0F2F1C5
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Insert
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 6

     Column Name = ACCTG_RULE_CD
     Value       = A

     Column Name = BAL_TYPE_CD
     Value       = IN

     Column Name = BAL_DATE
     Value       = 2006-02-16
```

```
    Column Name = BAL_AMOUNT
    Value       = 100.0
```

Conflict action of ignore

Example B-8 shows the output of the Exception Table Formatter for this exception when the conflict action is ignore. The only difference with Example B-7 on page 215 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore.

Example: B-8   Insert independent /parent row and a row with the same key exists - IGNORE

```
Exception Time     = 2006-02-16 17:27:28.548687
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE60130C97560001'
  SRC_TRANS_TIME     = 2006-02-16 22:25:30.591153
  SUBNAME            = BAL_V10002
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = BALRV1RP?0000000206
             (ASCII) = ????†¤??¯???????o?"
               (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F0F2F0F6
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Row Operation      = Insert
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 6

     Column Name = ACCTG_RULE_CD
     Value       = A

     Column Name = BAL_TYPE_CD
     Value       = IN

     Column Name = BAL_DATE
     Value       = 2006-02-16

     Column Name = BAL_AMOUNT
     Value       = 100.0
```

B.2.1.2   Insert child and the same key and child’s parents exists

This scenario corresponds to the case where a propagated row that is a child row not only finds a row with the corresponding key but also discovers that the parent row of the propagated row exists on the same server. The column values in the non-key columns of the propagated row may or may not match the corresponding non-key column values in the existing row. An exception is generated and the propagated row may either be ignored (when the conflict action is ignore) or overwrite the existing row when the conflict action is force. Any conflict between any non-key values in the propagated row and that of the existing row are not identified.

Conflict action of force

Example B-9 shows the output of the Exception Table Formatter for this exception when the conflict action is force. It describes the “Row Operation” being an ‘Insert,’ which has an exception with a “REASON” of ‘Duplicate’, “SQL CODE” of ‘-803,’ and SQLSTATE” of

‘23505’. It identifies the “CONFLICT_RULE” as being ‘C’, and the conflict action “IS_APPLIED” of ‘Y,’ which corresponds to a force. The values in the overriding row are shown in the Columns portion of the output. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field). Also, no information is provided whether or not there were mismatches in the non-key columns between the propagated row and the existing row.

Note: This is the action regardless of whether the delete rule is CASCADE or RESTRICT for the dependent tables.

Example: B-9 Insert child row and row with the same key and its parent exists - FORCE

```
Exception Time     = 2006-02-16 17:27:39.123015
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000007C4DD090000'
  SRC_TRANS_TIME     = 2006-02-16 22:24:54.300215
  SUBNAME            = BAL_V10001
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = BALRV1RP?000000021E
             (ASCII) = ????†¤??¯???????o¤
               (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F0F2F1C5
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Insert
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 6

     Column Name = ACCTG_RULE_CD
     Value       = A

     Column Name = BAL_TYPE_CD
     Value       = IN

     Column Name = BAL_DATE
     Value       = 2006-02-16

     Column Name = BAL_AMOUNT
     Value       = 100.0
```

Conflict action of ignore

Example B-10 shows the output of the Exception Table Formatter for this exception when the conflict action is ignore. The only difference from Example B-9 on page 217 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore.

Example: B-10 Insert child row and the same key and child’s parents exists - IGNORE

```
Exception Time     = 2006-02-16 17:27:28.548687
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE60130C97560001'
  SRC_TRANS_TIME     = 2006-02-16 22:25:30.591153
  SUBNAME            = BAL_V10002
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = BALRV1RP?0000000206
             (ASCII) = ????†¤??¯???????o?"
               (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F0F2F0F6
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Row Operation      = Insert
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 6

     Column Name = ACCTG_RULE_CD
     Value       = A

     Column Name = BAL_TYPE_CD
     Value       = IN

     Column Name = BAL_DATE
     Value       = 2006-02-16

     Column Name = BAL_AMOUNT
     Value       = 100.0
```

B.2.1.3 Insert child but same key and child's parent missing

This scenario corresponds to the case where a propagated row that is a child row not only does not find a row with the corresponding key but also discovers that the parent row of the propagated row is also missing.

Note: No conflict exception is generated and Q Apply attempts to insert the row. Therefore, the conflict action rule of force/ignore does not apply in this case.

However, an SQL error exception is generated and the propagated row fails to insert the child row with a referential constraint violation SQL CODE -530 indicating that a parent row was missing. Example B-11 shows the output of the Exception Table Formatter for this SQL error exception. It describes the "Row Operation" being an 'Insert,' which has an exception with a "REASON" of 'OKSQLSTATE', "SQL CODE" of '-530,' and SQLSTATE" of '23503'. The 'OKSQLSTATE' indicates that the '23503' state was recorded in the OKSQLSTATES column of the IBMQREP_TARGETS table for this subscription, and therefore the SQL error action is ignored by the Q Apply program. The "SQLCODE" value of '-530' corresponds to a missing parent row. Because of the failure to insert the row successfully, the conflict action "IS_APPLIED" shows 'N'. The values in the overriding row are shown in the "Columns" portion of the output. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field).

Example: B-11 Insert child row but same key and child's parent missing

```
Exception Time     = 2006-03-07 21:12:43.719295
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'000000002591E1B00000'
  SRC_TRANS_TIME     = 2006-03-08 02:12:26.064894
  SUBNAME            = PROD_V10001
  REASON             = OKSQLSTATE
  SQLCODE            = -530
  SQLSTATE           = 23503
```

```
SQLERRMC  (EBCDIC) = FK_PROD_V1
           (ASCII) = '?m??Žm†¤
             (HEX) = C6D26DD7D9D6C46DE5F1
IS_APPLIED         = N
CONFLICT_RULE      = C
Database name      = DB8G
Row Operation      = Insert
Columns :
   Column Name = ACCT_ID
   Is Key      = true
   Value       = 1206

   Column Name = PROD_TYPE
   Is Key      = true
   Value       = A

   Column Name = PROD_DATE
   Value       = 2006-03-02

   Column Name = PROD_QTY
   Value       = 100
```

B.2.1.4 Insert child & same key exists but child's new parents missing

This scenario corresponds to the case where a propagated row that is a child row finds a row with the corresponding key[3], but discovers that a parent row of the propagated row is missing.

When the conflict action is ignore, one conflict exception for finding a duplicate child row is generated. When the conflict action is force, two exceptions are generated: one for finding the duplicate child row, and the other for an SQL error exception (referential constraint violation SQL CODE -530 indicating that a parent row was missing) for a missing parent row. Any conflicts between any non-key values in the propagated row and that of the existing row are not identified.

Conflict action of force

The two exceptions generated for this scenario with the conflict action of force follow:

- Example B-12 shows the output of the Exception Table Formatter for the conflict exception. It describes the "Row Operation" being an 'Insert,' which has an exception with a "REASON" of 'Duplicate', "SQL CODE" of '-803,' and SQLSTATE" of '23505'. It identifies the "CONFLICT_RULE" as being 'C', and the conflict action "IS_APPLIED" of 'Y,' which corresponds to a force. The values in the overriding row are shown in the Columns portion of the output. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field). Also, no information is provided whether or not there were mismatches in the non-key columns between the propagated row and the existing row.

> **Note:** Even though the "IS_APPLIED" field has a value of 'Y', it does not necessarily indicate that the propagated row was applied successfully. It only indicates that Q Apply intends to apply the change. You have to check for an SQL error exception to determine whether the propagated row was actually applied successfully. The correlating fields are SRC_COMMIT_LSN and SRC_TRANS_TIME.

Example: B-12 Insert child row but same key and child's parent is missing - FORCE 1/2

```
Exception Time     = 2006-03-07 22:18:24.325188
```

[3] the column values in the non-key columns of the propagated row may or may not match the corresponding non-key column values in the existing row.

```
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'00000000259D7AF10000'
  SRC_TRANS_TIME     = 2006-03-08 03:15:14.565397
  SUBNAME            = PROD_V20001
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = PRODRV2R?0000000201
             (ASCII) = ??Ž?†o?¯???????o?¤
               (HEX) = D7D9D6C4D9E5F2D9FFF0F0F0F0F0F0F0F2F0F1
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Insert
  Columns :
     Column Name = PROD_ID
     Is Key      = true
     Value       = 100

     Column Name = ACCT_ID
     Value       = 1252

     Column Name = PROD_QTY
     Value       = 100
```

- Example B-13 shows the output of the Exception Table Formatter for the SQL error exception. It describes the “Row Operation” being an ‘Insert,’ which has an exception with a “REASON” of ‘OKSQLSTATE’, “SQL CODE” of ‘-530,’ and SQLSTATE” of ‘23503’. The ‘OKSQLSTATE’ indicates that the ‘23503’ state was recorded in the OKSQLSTATES column of the IBMQREP_TARGETS table for this subscription, and therefore the SQL error action is ignored by the Q Apply program. The “SQLCODE” value of ‘-530’ corresponds to a missing parent row. Because of the failure to insert the row successfully, the conflict action “IS_APPLIED” shows ‘N’. The values in the overriding row are shown in the Columns portion of the output. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field).

Example: B-13 Insert child row but same key and child’s parent is missing - FORCE 2/2

```
Exception Time     = 2006-03-07 22:18:24.338169
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'00000000259D7AF10000'
  SRC_TRANS_TIME     = 2006-03-08 03:15:14.565397
  SUBNAME            = PROD_V20001
  REASON             = OKSQLSTATE
  SQLCODE            = -530
  SQLSTATE           = 23503
  SQLERRMC  (EBCDIC) = FK_PROD_V2
             (ASCII) = '?m??Žm†o
               (HEX) = C6D26DD7D9D6C46DE5F2
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Insert
  Columns :
     Column Name = PROD_ID
     Is Key      = true
     Value       = 100

     Column Name = ACCT_ID
     Value       = 1252
```

```
    Column Name = PROD_QTY
    Value       = 100
```

Conflict action of ignore

Example B-14 shows the output of the Exception Table Formatter for this conflict exception when the conflict action is ignore. The only difference from Example B-12 on page 219 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore. As mentioned earlier, no SQL error exception is generated.

Example: B-14   Insert child row but same key and child’s parent is missing - IGNORE

```
Exception Time     = 2006-03-07 21:23:43.811586
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000000025929C5E0000'
  SRC_TRANS_TIME     = 2006-03-08 02:23:20.902584
  SUBNAME            = PROD_V10001
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = PRODRV1R?0000000201
             (ASCII) = ??Ž?†¤?¯??????o?¤
               (HEX) = D7D9D6C4D9E5F1D9FFF0F0F0F0F0F0F0F2F0F1
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Row Operation      = Insert
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 1200

     Column Name = PROD_TYPE
     Is Key      = true
     Value       = A

     Column Name = PROD_DATE
     Value       = 2006-03-02

     Column Name = PROD_QTY
     Value       = 10
```

B.2.2 Exceptions with deletes

In this section we describe exceptions generated by a delete operation propagated from one server to another.

B.2.2.1 Delete independent/parent/child and same key missing

This scenario corresponds to the case where a propagated delete operation of an independent/parent/child row does not find an existing row with the same key. An exception is generated and the propagated delete operation is assumed to have successfully deleted the existing row.

Conflict action is force

Example B-15 shows the output of the Exception Table Formatter for this exception when the conflict action is force. It describes the “Row Operation” being an ‘Delete,’ which has an exception with a “REASON” of ‘NOT FOUND’, “SQL CODE” of ‘100,’ and SQLSTATE” of ‘02000’. It identifies the “CONFLICT_RULE” as being ‘C’, and the conflict action “IS_APPLIED” of ‘Y,’ which corresponds to a force. The columns section shows the key of the propagated delete operation.

Example: B-15 Delete independent/parent/child row and same key missing - FORCE

```
Exception Time     = 2006-02-16 13:30:44.142004
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000007B85C090000'
  SRC_TRANS_TIME     = 2006-02-16 18:29:56.577283
  SUBNAME            = BAL_V10001
  REASON             = NOTFOUND
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Delete
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 101
```

Conflict action is ignore

Example B-16 shows the output of the Exception Table Formatter for this conflict exception when the conflict action is ignore. The only difference from Example B-15 on page 222 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore.

Example: B-16 Delete independent/parent/child row and same key missing - IGNORE

```
Exception Time     = 2006-02-16 17:08:04.335195
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE600EBFB2740001'
  SRC_TRANS_TIME     = 2006-02-16 22:06:16.697079
  SUBNAME            = BAL_V10002
  REASON             = NOTFOUND
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Source Table Owner = null
  Source Table Name  = null
  Row Operation      = Delete
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 109
```

B.2.2.2 Delete independent and same key exists

This scenario corresponds to the case where a propagated delete operation of an independent row finds an existing row with the same key. The column values in the non-key columns of the propagated delete operation may or may not match the corresponding non-key column values in the existing row. If the non-key column values in the propagated row do not match one or more corresponding non-key column values in the existing row, an exception is generated. When the conflict action is force, the existing row is deleted. When the conflict action is ignore, the existing row is not deleted.

Conflict action is force

Example B-17 shows the output of the Exception Table Formatter for this exception when the conflict action is force. It describes the “Row Operation” being an ‘Delete,’ which has an exception with a “REASON” of ‘CHECKFAILED’, “SQL CODE” of ‘100,’ and SQLSTATE” of ‘02000’. It identifies the “CONFLICT_RULE” as being ‘C’, and the conflict action “IS_APPLIED” of ‘Y,’ which corresponds to a force. The columns section shows the key of the propagated row.

Example: B-17 Delete independent row and same key exists

```
Exception Time     = 2006-02-16 17:08:08.069676
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000007C28D700000'
  SRC_TRANS_TIME     = 2006-02-16 22:05:55.729733
  SUBNAME            = BAL_V10001
  REASON             = CHECKFAILED
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Delete
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 110
```

Conflict action is ignore

Example B-18 shows the output of the Exception Table Formatter for this conflict exception when the conflict action is ignore. The only difference from Example B-17 on page 223 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore.

Example: B-18 Delete independent row and same key exists

```
Exception Time     = 2006-02-16 13:30:37.554809
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE5FDE608B6C0001'
  SRC_TRANS_TIME     = 2006-02-16 18:29:52.544614
  SUBNAME            = BAL_V10002
  REASON             = CHECKFAILED
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Row Operation      = Delete
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 103
```

B.2.2.3 Delete parent and same key exists

This scenario corresponds to the case where a propagated delete operation of a parent row finds an existing row with the same key. The column values in the non-key columns of the propagated delete operation may or may not match the corresponding non-key column values in the existing row. If the non-key column values in the propagated row do not match one or more corresponding non-key column values in the existing row, an exception is generated. The propagated delete operation may either be ignored (when the conflict action is ignore) or attempted on the existing row when the conflict action is force. If the delete operation should fail for any reason such as a DELETE RESTRICT referential constraint violation, an SQL error exception is also generated.

Conflict action is force

Assuming that the non-key column values in the propagated row do not match one or more corresponding non-key column values in the existing row, and an SQL error occurs when the delete operation is attempted on the existing row, the two exceptions generated for this scenario with the conflict action of force follows:

- Example B-19 shows the output of the Exception Table Formatter for the conflict exception. It describes the “Row Operation” being an ‘Delete,’ which has an exception with a “REASON” of ‘CHECKFAILED’, “SQL CODE” of ‘100,’ and SQLSTATE” of ‘02000’. It identifies the “CONFLICT_RULE” as being ‘C’, and the conflict action “IS_APPLIED” of ‘Y,’ which corresponds to a force. The key of the propagated delete operation is shown in the Columns portion of the output. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field).

  Note: Even though the “IS_APPLIED” field has a value of ‘Y’, it does not necessarily indicate that the propagated row was applied successfully. It only indicates that Q Apply intends to apply the change. You have to check for an SQL error exception to determine whether the propagated row was actually applied successfully. The correlating fields are SRC_COMMIT_LSN and SRC_TRANS_TIME.

Example: B-19 Delete parent row and same key exists - value-based conflict - FORCE

```
Exception Time     = 2006-03-07 23:03:56.486056
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000025A2860D0000'
  SRC_TRANS_TIME     = 2006-03-08 04:03:23.935969
  SUBNAME            = BAL_V10001
  REASON             = CHECKFAILED
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Delete
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4002
```

- Example B-20 shows the output of the Exception Table Formatter for the SQL error exception. It describes the “Row Operation” being an ‘Delete,’ which has an exception with a “REASON” of ‘OKSQLSTATE’, “SQL CODE” of ‘-532,’ and SQLSTATE” of ‘23504’. The ‘OKSQLSTATE’ indicates that the ‘23504’ state was recorded in the OKSQLSTATES column of the IBMQREP_TARGETS table for this subscription, and therefore the SQL error action is ignored by the Q Apply program. The “SQLCODE” value of ‘-532’ corresponds to a referential constraint violation due to a DELETE RESTRICT rule. Because of the failure to delete the parent row successfully, the conflict action “IS_APPLIED” shows ‘N’.

Example: B-20   Delete parent row and same key exists - SQL error exception

```
Exception Time     = 2006-03-07 23:03:56.503344
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000025A2860D0000'
  SRC_TRANS_TIME     = 2006-03-08 04:03:23.935969
  SUBNAME            = BAL_V10001
  REASON             = OKSQLSTATE
  SQLCODE            = -532
  SQLSTATE           = 23504
  SQLERRMC  (EBCDIC) = FK_PROD_V1?0000000202
             (ASCII) = '?m??Žm†¤¯???????o?o
               (HEX) = C6D26DD7D9D6C46DE5F1FFF0F0F0F0F0F0F0F2F0F2
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Delete
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4002
```

Conflict action is ignore

Example B-21 shows the output of the Exception Table Formatter for this conflict exception when the conflict action is ignore. The only difference from Example B-19 on page 224 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore. As mentioned earlier, no SQL error exception is generated.

Example: B-21   Delete parent row and same key exists - value-based conflict - IGNORE

```
Exception Time     = 2006-03-07 22:43:13.8796
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE783D767B180001'
  SRC_TRANS_TIME     = 2006-03-08 03:43:03.480833
  SUBNAME            = BAL_V10002
  REASON             = CHECKFAILED
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Row Operation      = Delete
```

```
Columns :
   Column Name = ACCT_ID
   Is Key      = true
   Value       = 4005
```

B.2.3 Exceptions with updates

In this section, we describe exceptions generated by the propagation of an update operation from one server to the other. The scenarios include updates where the propagated update includes the key and non-key columns, only the non-key columns, or key columns only.

B.2.3.1 Update independent/parent/child and key missing

This scenario corresponds to the case where a propagated update operation of an independent/parent/child row does not find an existing row with the same key. An exception is generated. If the conflict action is force, then the update operation is converted to an insert operation and the insert is attempted. This may or may not result in an additional SQL error exception. If the conflict option is ignore, the update operation is ignored altogether with no conversion to an insert.

Conflict action is force

Example B-22 shows the output of the Exception Table Formatter for this exception when the conflict action is force. It describes the “Row Operation” being an ‘Update,’ which has an exception with a “REASON” of ‘NOT FOUND’, “SQL CODE” of ‘100,’ and SQLSTATE” of ‘02000’. It identifies the “CONFLICT_RULE” as being ‘C’, and the conflict action “IS_APPLIED” of ‘Y,’ which corresponds to a force. The columns section shows the values of all the columns (both key and non-key) key of the propagated update operation.

Note: Even though the “IS_APPLIED” field has a value of ‘Y’, it does not necessarily indicate that the propagated row was inserted successfully. It only indicates that Q Apply intends to insert the row. You have to check for an SQL error exception to determine whether the propagated row was actually inserted successfully. The correlating fields are SRC_COMMIT_LSN and SRC_TRANS_TIME.

The assumption here is that the converted insert operation is successful. If the insert fails, an SQL error exception would also be generated.

Example: B-22 Update independent/parent/child row and same key missing - FORCE

```
Exception Time     = 2006-03-09 12:06:57.059223
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'000000002B28E6090000'
  SRC_TRANS_TIME     = 2006-03-09 17:06:51.177977
  SUBNAME            = PROD_V10001
  REASON             = NOTFOUND
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Update
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
```

```
    Value       = 4140
    Before Value= 4041

    Column Name = PROD_TYPE
    Is Key      = true
    Value       = T
    Before Value= A

    Column Name = PROD_DATE
    Value       = 2006-03-09
    Before Value= 2006-03-02

    Column Name = PROD_QTY
    Value       = 100
```

Conflict action is ignore

Example B-23 shows the output of the Exception Table Formatter for this conflict exception when the conflict action is ignore. The only difference with Example B-22 on page 226 is that the conflict action "IS_APPLIED" is 'N,' which corresponds to an ignore.

Example: B-23 Update independent/parent/child row and same key missing - IGNORE

```
Exception Time     = 2006-03-09 12:06:42.447877
  RECVQ              = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN     = x'0000BE7A32F248F80001'
  SRC_TRANS_TIME     = 2006-03-09 17:06:39.285032
  SUBNAME            = PROD_V10002
  REASON             = NOTFOUND
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8A
  Row Operation      = Update
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4141
     Before Value= 4040

     Column Name = PROD_TYPE
     Is Key      = true
     Value       = S
     Before Value= A

     Column Name = PROD_DATE
     Value       = 2006-03-09
     Before Value= 2006-03-02
```

B.2.3.2 Update independent/parent/child and key exists - mismatch

This scenario corresponds to the case where a propagated update operation of an independent/parent/child row finds an existing row with the same key. The column values in the non-key columns of the propagated update operation may or may not match the corresponding non-key column values in the existing row. The propagated update operation may either be ignored (when the conflict action is ignore) or attempted on the existing row when the conflict action is force. If the update operation should fail for any reason such as an UPDATE RESTRICT referential constraint violation or a missing parent row, an SQL error exception is also generated.

Conflict action is force

Assuming that the non-key column values in the propagated row do not match one or more corresponding non-key column values in the existing row, and an SQL error occurs when the update operation is attempted on the existing row, the two exceptions generated for this scenario with the conflict action of force follow:

- Example B-24 shows the output of the Exception Table Formatter for the conflict exception. It describes the "Row Operation" being an 'Update,' which has an exception with a "REASON" of 'CHECKFAILED', "SQL CODE" of '100,' and SQLSTATE" of '02000'. It identifies the "CONFLICT_RULE" as being 'C', and the conflict action "IS_APPLIED" of 'Y,' which corresponds to a force. The Columns portion of the output shows all the column values (which it needs in case Q Apply needs to convert the update to an insert operation), including the before values of the columns that are updated in order to be able to check for value-based conflicts. The name of the table involved in the exception needs to be derived from the subscription name (SUBNAME field).

> **Note:** Even though the "IS_APPLIED" field has a value of 'Y', it does not necessarily indicate that the propagated row was applied successfully. It only indicates that Q Apply intends to apply the change. You have to check for an SQL error exception to determine whether the propagated row was actually applied successfully. The correlating fields are SRC_COMMIT_LSN and SRC_TRANS_TIME.

Example: B-24 Update independent/parent/child row and same key exists - value-based conflict - FORCE

```
Exception Time     = 2006-03-09 14:03:39.189442
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'000000002B34C7480000'
  SRC_TRANS_TIME     = 2006-03-09 19:03:07.601041
  SUBNAME            = PROD_V10001
  REASON             = CHECKFAILED
  SQLCODE            = 100
  SQLSTATE           = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED         = Y
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Update
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4030
     Before Value= 4003

     Column Name = PROD_TYPE
     Is Key      = true
     Value       = H
     Before Value= Z

     Column Name = PROD_DATE
     Value       = 2006-03-09
     Before Value= 2006-03-01
```

```
   Column Name = PROD_QTY
   Value       = 100
```

- Example B-25 shows the output of the Exception Table Formatter for the SQL error exception. It describes the “Row Operation” being an ‘Update,’ which has an exception with a “REASON” of ‘OKSQLSTATE’, “SQL CODE” of ‘-530,’ and SQLSTATE” of ‘23503’. The ‘OKSQLSTATE’ indicates that the ‘23503’ state was recorded in the OKSQLSTATES column of the IBMQREP_TARGETS table for this subscription, and therefore the SQL error action is ignored by the Q Apply program. The “SQLCODE” value of ‘-530’ corresponds to a referential constraint violation due to a missing parent row. Because of the failure to update the child row successfully, the conflict action “IS_APPLIED” shows ‘N’.

Example: B-25 Update independent/parent/child row and same key exists - SQL error exception

```
Exception Time     = 2006-03-09 14:03:39.202276
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'000000002B34C7480000'
  SRC_TRANS_TIME     = 2006-03-09 19:03:07.601041
  SUBNAME            = PROD_V10001
  REASON             = OKSQLSTATE
  SQLCODE            = -530
  SQLSTATE           = 23503
  SQLERRMC  (EBCDIC) = FK_PROD_V1
             (ASCII) = '?m??Žm†¤
               (HEX) = C6D26DD7D9D6C46DE5F1
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Update
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4030
     Before Value= 4003

     Column Name = PROD_TYPE
     Is Key      = true
     Value       = H
     Before Value= Z

     Column Name = PROD_DATE
     Value       = 2006-03-09
     Before Value= 2006-03-01

     Column Name = PROD_QTY
     Value       = 100
```

Conflict action is ignore

Example B-26 shows the output of the Exception Table Formatter for this conflict exception when the conflict action is ignore. The only difference from Example B-24 on page 228 is that the conflict action “IS_APPLIED” is ‘N,’ which corresponds to an ignore. As mentioned earlier, no SQL error exception is generated.

Example: B-26   Update independent/parent/child row and same key exists, value-based conflict, IGNORE

```
Exception Time     = 2006-03-09 12:51:54.241501
  RECVQ             = QREP.POKA.TO.POKB.RECVQ
  SRC_COMMIT_LSN    = x'0000BE7A3D0815D10001'
  SRC_TRANS_TIME    = 2006-03-09 17:51:47.085218
  SUBNAME           = PROD_V10002
  REASON            = CHECKFAILED
  SQLCODE           = 100
  SQLSTATE          = 02000
  SQLERRMC  (EBCDIC) =
             (ASCII) =
               (HEX) =
  IS_APPLIED        = N
  CONFLICT_RULE     = C
  Database name     = DB8A
  Row Operation     = Update
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4141
     Before Value= 4000

     Column Name = PROD_TYPE
     Is Key      = true
     Value       = X
     Before Value= A

     Column Name = PROD_DATE
     Value       = 2006-03-09
     Before Value= 2006-03-02
```

B.2.3.3   Update independent/parent/child and key exists - all match

This scenario also corresponds to the case where a propagated update operation of an independent/parent/child row finds an existing row with the same key. The column values in the non-key columns of the propagated update operation may or may not match the corresponding non-key column values in the existing row. The propagated update operation may either be ignored (when the conflict action is ignore) or attempted on the existing row when the conflict action is force. If the update operation should fail for any reason such as an UPDATE RESTRICT referential constraint violation or a missing parent row, an SQL error occurs.

Note: In this scenario, all the non-key column values in the propagated row match all the corresponding column values in the existing row, and therefore no conflict exception is generated. Since there is no conflict exception, the conflict rule is ignored.

The Q Apply program proceeds with updating the existing row with the updates identified in the propagated row. However, this update operation may fail due to SQL errors resulting from referential constraint or uniqueness constraint violations.

Example B-27 shows the output of the Exception Table Formatter for an SQL error exception that occurs without a corresponding conflict exception. The error encountered was that the updated key value of 4021 (updated from 4020) already exists. It describes the “Row Operation” being an ‘Update,’ which has an exception with a “REASON” of ‘DUPLICATE’, “SQL CODE” of ‘-803,’ and SQLSTATE” of ‘23503’. Because of the failure to update the existing row successfully, the conflict action “IS_APPLIED” shows ‘N’.

Example: B-27   Update independent/parent/child row and key exists - all match - SQL error

```
Exception Time     = 2006-03-08 12:43:34.632773
  RECVQ              = QREP.POKB.TO.POKA.RECVQ
  SRC_COMMIT_LSN     = x'0000000025CA5AFE0000'
  SRC_TRANS_TIME     = 2006-03-08 17:43:20.137602
  SUBNAME            = BAL_V10001
  REASON             = DUPLICATE
  SQLCODE            = -803
  SQLSTATE           = 23505
  SQLERRMC  (EBCDIC) = BALRV1RP?000000020E
             (ASCII) = ????†¤??¯???????o?
               (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F0F2F0C5
  IS_APPLIED         = N
  CONFLICT_RULE      = C
  Database name      = DB8G
  Row Operation      = Update
  Columns :
     Column Name = ACCT_ID
     Is Key      = true
     Value       = 4021
     Before Value= 4020

     Column Name = ACCTG_RULE_CD
     Value       = A

     Column Name = BAL_TYPE_CD
     Value       = IN

     Column Name = BAL_DATE
     Value       = 2006-03-08

     Column Name = BAL_AMOUNT
     Value       = 400.0
```

Note: The update of only the key columns in the propagated row is a special case of this scenario.

B.3 Data inconsistencies in an HA bidirectional scenario

In this section we describe a few scenarios that result in data inconsistencies between the source and target, and analyze the exceptions generated when such conditions occur. In a few cases, data inconsistencies occur but the exceptions recorded in the IBMQREP_EXCEPTIONS table do not provide sufficient information to troubleshoot such occurrences.

Very important: Data inconsistencies described in this section are not specific to failover and switchback operations in a bidirectional environment. Similar data inconsistencies can occur during normal operations when the bidirectional environment is operated in Active/Active mode. Both the source and the target are updatable during normal operations. We strongly recommend that bidirectional replication scenarios be implemented in a way that eliminates conflicts altogether. For implementations requiring a more robust conflict resolution mechanism, we recommend a peer-to-peer replication implementation.

Figure B-18 shows the tables used to demonstrate the data inconsistencies scenarios.

- The independent tables ST on the source and TT on the target have no referential constraints and are used in the non-referential integrity data inconsistency scenarios described in B.3.1, “Non-referential integrity-related data inconsistency scenarios” on page 233.
- The independent tables SIC on the source and TIC on the target have no referential constraints, but have their key column defined as an IDENTITY data type. These tables are also used in the non-referential integrity data inconsistency scenarios described in B.3.1, “Non-referential integrity-related data inconsistency scenarios” on page 233.
- The tables SPTA and SCTA have a referential relationship with a delete rule of RESTRICT on the source, while table TPTA and TCTA are their corresponding tables on the target. These tables are used in the referential integrity data inconsistency scenarios described in B.3.2, “Referential integrity-related data inconsistency scenarios” on page 242.

As mentioned earlier, for the bidirectional high availability scenario described in this book, our recommendations for the conflict rule, conflict action, and error action are as follows:

- Conflict rule of ‘C’ (key columns and changed columns) for both the source and target subscriptions.
- Conflict action of ‘F’ (force) for the source subscription.
- Conflict action of ‘I’ (ignore) for the target subscription.
- Add the sqlstates 23001, 23503, 23504, and 23505 to the OKSQLSTATES column in the IBMQREP_TARGETS tables on both the source and target subscriptions.
- Error action of ‘Q’ (stop reading from the corresponding receive queue) on both the source and target subscriptions.

These specifications are assumed in the scenarios described.

Attention: The scenarios specify SQL operations such as inserts, updates, and deletes occurring at the source (prior to failover) and target (after failover but prior to switchback). Each SQL operation is considered to be a separate transaction. To reiterate, user updates occur only at the source (prior to failover) or at the target (after failover and prior to switchback). Therefore no conflicts will occur during pre-failover and post-failover operations. However, during switchback operations when resynchronization of source and target data occurs, conflicts are most likely to occur. It is assumed that during switchback, the Q Capture and Q Apply programs are running on both the source and target resulting in unpropagated changes on the source and target being propagated to the other server.

The data inconsistency scenarios are broadly classified into two categories—those involving tables with no referential relationships, and those with referential relationships.

Note: The data inconsistency scenarios described here are not necessarily a complete list of all possible data inconsistency cases. You are advised to use these scenarios to achieve a better understanding of the process of conflict action and error action in bidirectional Q replication environments.

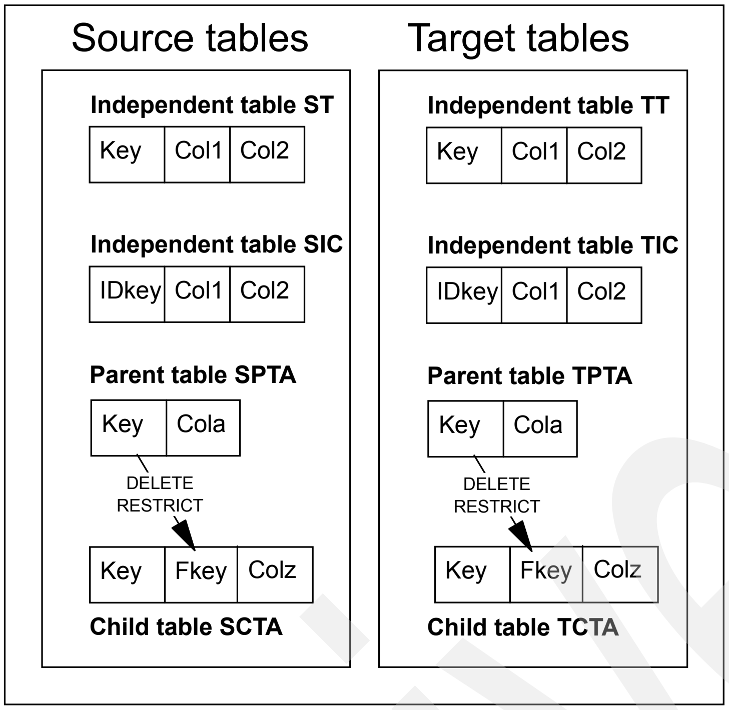


Figure B-18 Tables used to demonstrate the data inconsistency scenarios

B.3.1 Non-referential integrity-related data inconsistency scenarios

Table B-2 provides a summary of the potential non-referential integrity-related data inconsistency scenarios that result in the data on the source and target becoming inconsistent at the conclusion of the switchback operation. Each scenario constitutes a simple set of operations on the source and target.

Table B-2 Non-referential integrity-related data inconsistent scenarios

S#	Unpropagated changes before failover	Unpropagated changed before switchback	Source data after switchback completed	Target data after switchback completed	Source exceptions: ► Conflict rule is ‘C’. ► Conflict action is FORCE. ► Error action is Q. ► OKSQLSTATES 23001 23503 23504.	Target exceptions: ► Conflict rule is ‘C’. ► Conflict action is FORCE. ► Error action is Q. ► OKSQLSTATES 23001 23503 23504.
1	Insert ST (3,A,100)	Insert TT (3,A,200) Delete TT (3)	Empty	TT (3,A,100)	Row Operation ‘Insert’ SQL Code ‘-803’ Reason ‘Duplicate’ IS_APPLIED ‘Y’	None
2	Insert ST (4,A,200) Delete ST (4)	Insert TT (4,A,200)	ST (4,A,200)	Empty	None	Row Operation ‘Inser’t SQL Code ‘-803’ Reason ‘Duplicate’ IS_APPLIED ‘N’
3	Insert ST (5,A,100) Update ST (5,A,200)	Insert TT (5,A,300) Delete TT (5)	Empty	TT (5,A,200)	Row Operation ‘Insert’ SQL Code ‘-803’ Reason ‘Duplicate’ IS_APPLIED ‘Y’	None

S#	Unpropagated changes before failover	Unpropagated changed before switchback	Source data after switchback completed	Target data after switchback completed	Source exceptions: ▸ Conflict rule is 'C'. ▸ Conflict action is FORCE. ▸ Error action is Q. ▸ OKSQLSTATES 23001 23503 23504.	Target exceptions: ▸ Conflict rule is 'C'. ▸ Conflict action is FORCE. ▸ Error action is Q. ▸ OKSQLSTATES 23001 23503 23504.
4	Insert ST (6,B,100) Update ST (6,B,200)	Insert TT (6,C,100)	ST (6,C,100)	TT (6,C,200)	Row Operation Insert SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'Y'	Row Operation 'Insert' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'N'
5	Insert ST (12,A,300) Delete ST (12)	Insert TT (12,A,100) Update TT (12,A,300)	ST (12,A,300)	Empty	None	Row Operation 'Insert' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'N'
6	Insert ST (14,A,200) Update ST (14,A,100) Delete ST (14)	Insert TT (14,A,200) Update TT (14,A,100)	ST (14,A,100)	Empty	None	Row Operation 'Insert' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'N' Row Operation 'Update' SQL Code '100' Reason 'CHECKFAILED' IS_APPLIED 'N'
7	Existing row ST (94,A,100) Update key to 95	Existing row TT (94,A,100) Insert TT (95,B,200)	ST (95,B,200)	TT (94,A,100) TT (95,B,200)	Row Operation 'Insert' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'Y'	Row Operation 'Update' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'N'
8	Existing row ST (104,A,100) Delete ST (104)	Existing row TT(104,A,100) Delete TT (104) Insert TT (104,A,100)	ST (104,A,100)	Empty	Row Operation 'Delete' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'Y'	None
9	Existing row ST (107,C,300) Update key to 157	Existing row TT(107,C,300) Delete TT (107)	ST (157,C,300)	Empty	Row Operation 'Delete' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'Y'	Row Operation 'Update' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'N'
10	Existing row ST (109,A,100) Delete ST (109) Insert ST (109,A,100)	Existing row TT(109,A,100) Delete TT (109)	Empty	TT (109,A,100)	None	Row Operation 'Delete' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'N'
11	Existing row SIC (5,C,300) Update key to 7	Existing row TIC (5,C,300) Update key 5 to 8	SIC (7,C,300) SIC (8,C,300)	TIC (8,C,300)	Row Operation 'Update' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'Y'	Row Operation 'Update' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'N'
12	Existing row ST (92,A,100) Insert ST (93,B,200)	Existing row TT (92,A,100) Update key to 93	ST (92,A,100) ST (93,B,200)	TT (93,A,100)	Row Operation 'Update' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'N'	Row Operation 'Insert' SQL Code '-803' Reason 'Duplicate' IS_APPLIED 'N'

A brief description of the columns in Table B-2 follows:

- Unpropagated changes before failover refers to all the committed updates occurring on the source that did not get propagated over to the target at the point of failover. These changes may not yet have been captured from the DB2 log by Q Capture, or may be have been processed by Q Capture and stored in the transmit queue waiting to be sent to the receive queue on the target by WebSphere MQ.

- Unpropagated changes before switchback refers to all he committed updates occurring on the target that did not get propagated over to the source before the switchback process was initiated and completed. These changes may not yet have been captured from the DB2 log by Q Capture, or may be have been processed by Q Capture and stored in the transmit queue waiting to be sent to the receive queue on the source by WebSphere MQ.
- Source data after switchback completed describes the content of the source table at the end of switchback operations.
- Target data after switchback completed describes the content of the target table at the end of switchback operations.
- Source exceptions describe the exceptions recorded in the IBMQREP_EXCEPTIONS table at the source for this scenario for the conflict rule, conflict action, error action, and OKSQLSTATES options chosen. Only a partial list of the contents of the row is shown here.
- Target exceptions describe the exceptions recorded in the IBMQREP_EXCEPTIONS table at the target for this scenario for the conflict rule, conflict action, error action, and OKSQLSTATES options chosen. Only a partial list of the contents of the row is shown here.

Each of the scenarios resulting in data inconsistencies summarized in Table B-2 are briefly described here:

1. Scenario 1
   a. An insert of a row (3,A,100) into the source table ST is committed but not propagated to the target when failover occurs.
   b. After failover, the same row (3,A,100) is inserted into the target table TT in a transaction. A subsequent transaction deletes this row in the target table TT. These two changes are not propagated to the source since the source is unavailable.
   c. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.
   d. Propagation from target to source:
      i. The insert operation transaction from the target table TT encounters an existing row in the source table ST and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is force, it overwrites the row in the source table ST with the row propagated from the target table TT. It records this force action by setting the IS_APPLIED value to ‘Y’.
      ii. The delete operation transaction from the target table TT encounters a row with the matching key and matching changed non-key columns in the source table ST, and deletes it. No exception is generated.
      iii. At this point, the source table ST is empty.
   e. Propagation from source to target
      i. The insert operation transaction from the source table ST does not encounter any row with the same key in the target table TT. Since there is no conflict, Q Apply inserts the row (3,A,100) into the target table TT and no exception is generated.
      ii. At this point, the target table TT has a row (3,A,100).
   f. The source and target tables are now out of sync.

2. Scenario 2
   a. An insert of a row (4,A,200) into the source table ST is committed by a transaction, followed by a delete of the same row in another transaction. Neither of these changes are propagated to the target when failover occurs.
   b. After failover, the same row (4,A,200) is inserted into the target table TT in one transaction. This change is not propagated to the source since the source is unavailable.
   c. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.
   d. Propagation from target to source:
      i. The insert operation transaction from the target table TT does not encounter any row with the same key in the source table ST. Since there is no conflict, Q Apply inserts the row (4,A,200) into the source table ST and no exception is generated.
      ii. At this point, the source table ST has a row (4,A,200).
   e. Propagation from source to target:
      i. The insert operation transaction from the source table ST encounters an existing row in the target table TT and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is ignore, it ignores the row propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to 'N'.
      ii. The delete operation transaction from the source table ST encounters a row with the matching key and matching changed non-key columns in the target table TT, and deletes it. No exception is generated.
      iii. At this point, the target table TT is empty.
   f. The source and target tables are now out of sync.
3. Scenario 3
   a. An insert of a row (5,A,100) into the source table ST is committed by a transaction, followed by an update of the same row's non-key column Col2 to 200 in another transaction. Neither of these changes are propagated to the target when failover occurs.
   b. After failover, a row with the same key but different values in the Col2 column (5,A,300) is inserted into the target table TT in one transaction, followed by a delete of the same row in another transaction. Neither of these changes are propagated to the source since the source is unavailable.
   c. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.
   d. Propagation from target to source:
      i. The insert operation transaction from the target table TT encounters an existing row in the source table ST and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is force, it overwrites the row in the source table ST with the row propagated from the target table TT. It records this force action by setting the IS_APPLIED value to 'Y'.
      ii. The delete operation transaction from the target table TT encounters a row with the matching key and matching changed non-key columns in the source table ST, and deletes it. No exception is generated.
      iii. At this point, the source table ST is empty.

   e. Propagation from source to target:
      i. The insert operation transaction from the source table ST does not encounter any row with the same key in the target table TT. Since there is no conflict, Q Apply inserts the row (5,A,100) into the target table TT and no exception is generated.
      ii. The update operation transaction from the source table ST encounters a row with the matching key and matching changed non-key columns in the target table TT, and updates the row. No exception is generated.
      iii. At this point, the target table TT has row (5,A,200).
   f. The source and target tables are now out of sync.

4. Scenario 4
   a. An insert of a row (6,B,100) into the source table ST is committed by a transaction, followed by an update of the same row's Col2 column to 200 in another transaction. Neither of these changes are propagated to the target when failover occurs.
   b. After failover, a row with the same key but different values in the Col1 column (6,C,100) is inserted into the target table TT in one transaction. This change is not propagated to the source since the source is unavailable.
   c. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.
   d. Propagation from target to source:
      i. The insert operation transaction from the target table TT encounters an existing row in the source table ST and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is force, it overwrites the row in the source table ST with the row propagated from the target table TT. It records this force action by setting the IS_APPLIED value to 'Y'.
      ii. At this point, the source table ST has row (6,C,100).
   e. Propagation from source to target:
      i. The insert operation transaction from the source table ST encounters an existing row in the target table TT and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is ignore, it ignores the row propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to 'N'.
      ii. The update operation transaction from the source table ST encounters a row with the matching key and matching changed non-key columns in the target table TT, and updates the appropriate non-key column. No exception is generated.
      iii. At this point, the target table TT has row (6,C,200).

      Note: Since the conflict rule is 'C', the updated column Col2 value of 200 in the source table ST is reflected in the target table TT. Since the Col1 column was *not* updated in the source table ST, there was no change value for this Col1 propagated to the target, thereby leaving the existing value for this column in the target table TT.

   f. The source and target tables are now out of sync.

5. Scenario 5
   a. An insert of a row (12,A,300) into the source table ST is committed by a transaction, followed by a delete of the same row in another transaction. Neither of these changes are propagated to the target when failover occurs.

b. After failover, a row with the same key but different values in the Col2 column (12,A,100) is inserted into the target table TT in one transaction, an update of the same row's Col2 column to 300 in another transaction. Neither of these changes are propagated to the source since the source is unavailable.

c. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

d. Propagation from target to source:

   i. The insert operation transaction from the target table TT does not encounter any row with the same key in the source table ST. Since there is no conflict, Q Apply inserts the row (12,A,100) into the source table ST and no exception is generated.

   ii. The update operation transaction from the target table TT encounters a row with the matching key and matching changed non-key columns in the source table ST, and updates the row with the Col2 column value of 300. No exception is generated.

   iii. At this point, the source table ST has row (12,A,300).

e. Propagation from source to target:

   i. The insert operation transaction from the source table ST encounters an existing row in the target table TT and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is ignore, it ignores the row propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to 'N'.

   ii. The delete operation transaction from the source table ST encounters a row with the matching key and matching changed non-key columns in the target table TT, and deletes it. No exception is generated.

   iii. At this point, the target table TT is empty.

f. The source and target tables are now out of sync.

6. Scenario 6

   a. An insert of a row (14,A,200) into the source table ST is committed by a transaction, followed by an update of the Col2 column to 100 for the same row in another transaction, followed by a delete of the same row in yet another transaction. Neither of these changes are propagated to the target when failover occurs.

   b. After failover, an identical row with the same key is inserted into the target table TT in one transaction, followed by an update of the same row's Col2 column to 100 in another transaction. Neither of these changes are propagated to the source since the source is unavailable.

   c. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   d. Propagation from target to source:

      i. The insert operation transaction from the target table TT does not encounter any row with the same key in the source table ST. Since there is no conflict, Q Apply inserts the row (14,A,200) into the source table ST and no exception is generated.

      ii. The update operation transaction from the target table TT encounters a row with the matching key and matching changed non-key columns in the source table ST, and updates the row with the Col2 column value to 100. No exception is generated.

      iii. At this point, the source table ST has row (14,A,100).

   e. Propagation from source to target:

      i. The insert operation transaction from the source table ST encounters an existing row in the target table TT and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is ignore, it ignores the row propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to ‘N’.

   ii. The update operation transaction from the source table ST encounters a row with a matching key in the target table TT, but finds a mismatch on Col2. The before value of the changed column Col2 was 200 in the source table ST, but the before value of Col2 was 100 in the target table TT. An exception is generated with an SQLCODE 100 and a reason of CHECKFAILED. Since the conflict action is ignore, it ignores the update operation propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to ‘N’.

   iii. The delete operation transaction from the source table ST encounters a row with the matching key and matching changed non-key columns in the target table TT, and deletes it. No exception is generated.

   iv. At this point, the target table TT is empty.

f. The source and target tables are now out of sync.

7. Scenario 7

   a. Row (94,A,100) exists in the source table ST and target table TT.

   b. The key of the existing row (94,A,100) in the source table ST is updated to another value of 95. This change is not propagated to the target when failover occurs.

   c. After failover, another row (95,B,200) is inserted into the target table TT. This change is not propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   e. Propagation from target to source:

      i. The insert operation transaction from the target table TT encounters an existing row in the source table ST and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is force, it overwrites the row in the source table ST with the row propagated from the target table TT. It records this force action by setting the IS_APPLIED value to ‘Y’.

      ii. At this point, the source table ST has row (95,B,200).

   f. Propagation from source to target:

      i. The update operation transaction from the source table ST encounters a row with a matching key and matching changed non-key columns in the target table TT, and attempts to update the key to a value of 95. However, since the key value of 95 already exists, Q Apply gets an SQLCODE -803 and the update statement fails. An exception is generated with an SQLCODE -803 indicating a duplicate. Since the update operation could not be completed successfully, the IS_APPLIED value is set to ‘N’.

      ii. At this point, the target table TT has two rows in it, (94,A,100) and (95,B,200).

   g. The source and target tables are now out of sync.

8. Scenario 8

   a. Row (104,A,100) exists in the source table ST and target table TT.

   b. The existing row (104,A,100) in the source table ST is deleted. This change is not propagated to the target when failover occurs.

   c. After failover, the existing row (104,A,100) in the target table TT is deleted in one transaction, followed by an insert of another row with the same values as the deleted row in another transaction. None of these changes are propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   e. Propagation from target to source:

      i. The delete operation transaction from the target table TT does not find an existing row in the source table ST and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is force, and the row to be deleted does not exist in the source table ST, it assumes the operation completed successfully and sets the IS_APPLIED value to ‘Y’.

      ii. The insert operation transaction from the target table TT does not encounter any row with the same key in the source table ST. Since there is no conflict, Q Apply inserts the row (104,A,100) into the source table ST and no exception is generated.

      iii. At this point, the source table ST has row (104,A,100).

   f. Propagation from source to target:

      i. The delete operation transaction from the source table ST encounters a row with a matching key and matching changed non-key columns in the target table TT, and deletes it. No exception is generated.

      ii. At this point, the target table TT is empty.

   g. The source and target tables are now out of sync.

9. Scenario 9

   a. Row (107,C,300) exists in the source table ST and target table TT.

   b. The key of the existing row (107,C,300) in the source table ST is updated to a value of 157. This change is not propagated to the target when failover occurs.

   c. After failover, the existing row (107,C,300) in the target table TT is deleted. This change is not propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   e. Propagation from target to source:

      i. The delete operation transaction from the target table TT does not find an existing row in the source table ST and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is force, and the row to be deleted does not exist in the source table ST. It assumes the operation completed successfully and sets the IS_APPLIED value to ‘Y’.

      ii. At this point, the source table ST has row (157,C,300).

   f. Propagation from source to target:

      i. The update operation transaction from the source table ST does not find an existing row in the target table TT and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is ignore, the IS_APPLIED value is set to ‘N’.

      ii. At this point, the target table TT is empty.

   g. The source and target tables are now out of sync.

10. Scenario 10

   a. Row (109,A,100) exists in the source table ST and target table TT.

b. The existing row (109,A,100) in the source table ST is deleted in one transaction, and then re-inserted with the same values (109,A,100) in a subsequent transaction. These changes are not propagated to the target when failover occurs.

c. After failover, the existing row (109,A,100) in the target table TT is deleted. This change is not propagated to the source since the source is unavailable.

d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

e. Propagation from target to source:

   i. The delete operation transaction from the target table TT finds a matching key and matching changed non-key columns row in the source table ST and deletes it. No exception is generated.

   ii. At this point, the source table ST is empty.

f. Propagation from source to target:

   i. The delete operation transaction from the source table ST does not find an existing row in the target table TT and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is ignore, the IS_APPLIED value is set to 'N'.

   ii. The following insert operation transaction from the source table ST does not find an existing row in the target table TT with the same key, and inserts the row (109,A,100) propagated from the source table ST into the target table TT. No exception is generated.

   iii. At this point, the target table TT has row (109,A,100).

g. The source and target tables are now out of sync.

11. Scenario 11

a. Row (5,C,300) exists in the source table SIC and target table TIC.

b. The key of the existing row (5,C,300) in the source table SIC is updated to a different value. The SIC table has an identity column as its key, which means that the new value is assigned automatically by the system—let us say to the value 7 in this scenario. This change is not propagated to the target when failover occurs.

Important: Our recommendation for identity key columns in a bidirectional environments is to have one of the servers generate odd numbered keys, while the other one generates even numbered keys. This ensures that in a conflict situation, two different rows that accidentally have the same generated identity key do not overwrite one another in a conflict action of force situation. This recommendation may, however, result in the same business entity being recorded as two different rows, as demonstrated in this scenario.

c. After failover, the key of the existing row (5,C,300) in the target table TIC is also updated to a different value. Let us say that the new value of the automatically generated key is now 8. This change is not propagated to the source since the source is unavailable.

d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

e. Propagation from target to source:

   i. The update operation transaction from the target table TIC does not find an existing row in the source table SIC and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is force, the update operation is converted to an insert operation and row is inserted successfully into the source table SIC and the IS_APPLIED value is set to ‘N’.

   ii. At this point, the source table SIC has rows (7,C,300) and (8,C,300).

f. Propagation from source to target:

   i. The update operation transaction from the source table SIC does not find an existing row in the target table TIC and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is ignore, the update operation is not converted to an insert operation, and the IS_APPLIED value is set to ‘N’.

   ii. At this point, the target table TIC has row (8,C,300).

g. The source and target tables are now out of sync.

12. Scenario 12

   a. Row (92,A,100) exists in the source table ST and target table TT.

   b. Row (93,B,200) is inserted into the source table ST and committed. This change is not propagated to the target when failover occurs.

   c. After failover, the key of the existing row (92,A,100) in the target table TT is updated to a new value 93. This change is not propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   e. Propagation from target to source:

      i. The update operation transaction from the target table TT finds a matching key and matching changed non-key columns row in the source table ST and attempts to update the key to a value of 93. However, since the key value of 93 already exists, Q Apply gets an SQLCODE -803 and the update statement fails. An exception is generated with an SQLCODE -803 indicating a duplicate. Since the update operation could not be completed successfully, the IS_APPLIED value is set to ‘N’.

      ii. At this point, the source table ST has rows (92,A,100) and (93,B,200).

   f. Propagation from source to target:

      i. The insert operation transaction from the source table ST encounters an existing row in the target table TT and records an exception with an SQLCODE -803 indicating a duplicate. Since the conflict action is ignore, it ignores the row propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to ‘N’.

      ii. At this point, the target table TT has row (93,A,100).

   g. The source and target tables are now out of sync.

B.3.2 Referential integrity-related data inconsistency scenarios

Table B-3 provides a summary of the potential referential integrity-related data inconsistency scenarios that result in the data on the source and target becoming inconsistent at the conclusion of the switchback operation. Each scenario constitutes a simple set of operations on the source and target. A brief description of the columns in Table B-3 is provided in Section B.3.1, “Non-referential integrity-related data inconsistency scenarios” on page 233.

Table B-3 Referential integrity-related data inconsistent scenarios

S#	Unpropagated changes before failover	Unpropagated changed before switchback	Source data after switchback completed	Target data after switchback completed	Source exceptions ▸ Conflict rule is 'C'. ▸ Conflict action is FORCE. ▸ Error action is Q. ▸ OKSQLSTATES 23001 23503 23504.	Target exceptions ▸ Conflict rule is 'C'. ▸ Conflict action is FORCE. ▸ Error action is Q. ▸ OKSQLSTATES 23001 23503 23504.
1	Existing rows SPTA (1,A) SCTA (1,1,C1) Delete SCTA (1) Delete SPTA (1)	Existing rows TPTA (1,A) TCTA (1,1,C1) Update TCTA key to 2	Empty	TPTA (1,A) TCTA (2,1,C1)	Row Operation 'Update' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'Y' Row Operation 'Update' SQL Code '-530' SQLSTATE 23503 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Delete' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'N' Row Operation 'Delete' SQL Code '-532' SQLSTATE 23001 Reason 'OKSQLSTATE' IS_APPLIED 'N'
2	Existing rows SPTA (2,B) SCTA (2,2,C2) Delete SCTA (2) Delete SPTA (2)	Existing rows TPTA (2,B) TCTA (2,2,C2) Update TCTA non-key Col2 to Z2	Empty	TPTA (2,B) TCTA (2,2,Z2)	Row Operation 'Update' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'Y' Row Operation 'Update' SQL Code '-530' SQLSTATE 23503 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Delete' SQL Code '100' Reason 'CHECKFAILED' IS_APPLIED 'N' Row Operation 'Delete' SQL Code '-532' SQLSTATE 23001 Reason 'OKSQLSTATE' IS_APPLIED 'N'
3	Existing rows SPTA (3,C) SCTA (3,3,C3) Delete SCTA (3) Delete SPTA (3)	Existing rows TPTA (3,C) TCTA (3,3,C3) Insert TCTA (31,3,C9)	Empty	TPTA (3,C) TCTA (31,3,C9)	Row Operation 'Insert' SQL Code '-530' SQLSTATE 23503 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Delete' SQL Code '-532' SQLSTATE 23001 Reason 'OKSQLSTATE' IS_APPLIED 'N'
4	Existing row SPTA (4,D) Update SPTA key to 99	Existing row TPTA (4,D) Insert TCTA (4,4,C4)	SPTA (99,D)	TPTA (4,D) TCTA (4,4,C4)	Row Operation 'Insert' SQL Code '-530' SQLSTATE 23503 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Update' SQL Code '-531' SQLSTATE 23504 Reason 'OKSQLSTATE' IS_APPLIED 'N'
5	Existing rows SPTA (5,A) SCTA (5,5,C5) Update SCTA key to 8	Existing rows TPTA (5,A) TCTA (5,5,C5) Delete TCTA (5) Delete TPTA (5)	SPTA (5,A) SCTA (8,5,C5)	Empty	Row Operation 'Delete' SQL Code '100' Reason 'CHECKFAILED' IS_APPLIED 'Y' Row Operation 'Delete' SQL Code '-532' SQLSTATE 23001 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Update' SQL Code '100' Reason 'NOTFOUND' IS_APPLIED 'N'
6	Existing rows SPTA (6,C) SCTA (6,6,C6) Insert SCTA (61,6,C9)	Existing rows TPTA (6,C) TCTA (6,6,C6) Delete TCTA (6) Delete TPTA (6)	SPTA (6,C) SCTA (61,6,C9)	Empty	Row Operation 'Delete' SQL Code '-532' SQLSTATE 23001 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Insert' SQL Code '-530' SQLSTATE 23503 Reason 'OKSQLSTATE' IS_APPLIED 'N'
7	Existing row SPTA (7,E) Insert SCTA (7,7,C7)	Existing row TPTA (7,E) Update TPTA key to 97	SPTA (7,E) SCTA (7,7,C7)	TPTA (97,E)	Row Operation 'Update' SQL Code '-531' SQLSTATE 23504 Reason 'OKSQLSTATE' IS_APPLIED 'N'	Row Operation 'Insert' SQL Code '-530' SQLSTATE 23503 Reason 'OKSQLSTATE' IS_APPLIED 'N'

Each of the scenarios resulting in data inconsistencies is summarized in Table B-3 and is briefly described here:

1. Scenario 1
   a. Parent row (1,A) and corresponding child row (1,1,C1) exist in tables SPTA and SCTA on the source, respectively. Identical row exist in tables TPTA and TCTA on the target.
   b. The existing child row (1,1,C1) in the source table SCTA is deleted first, followed by a delete of the parent row (1,A). These changes are not propagated to the target when failover occurs.
   c. After failover, the key of the existing child row (1,1,C1) is updated to a value of 2. This change is not propagated to the source since the source is unavailable.
   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.
   e. Propagation from target to source:
      i. The update operation from the target table TCTA does not find an existing row in the source table SCTA and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is force, the update operation is converted to an insert operation and an attempt is made to insert the row into the source table SCTA. The IS_APPLIED value is set to 'Y'.

         However, this insert operation fails because the corresponding parent row is missing. An SQL error exception is generated with SQLCODE -530 indicating a missing parent row. Since the SQLSTATE 23503 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.
      ii. At this point, the source tables SPTA and SCTA are empty.
   f. Propagation from source to target:
      i. The delete operation of the child row (1,1,C1) from the source table SCTA does not find an existing row in the target table TCTA and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is ignore, the IS_APPLIED value is set to 'N'.
      ii. The delete operation of the parent row (1,A) from the source table SPTA finds a matching key and matching changed non-key columns row in the target table TPTA and attempts to delete it. However, since the delete rule is RESTRICT and the parent row (1,A) in the target table TPTA has a child row (2,1,C1) in the target table TCTA, the delete operation fails. An SQL error exception is generated with an SQLCODE -532 indicating a referential constraint violation. Since the SQLSTATE 23001 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.
      iii. At this point, the target table TPTA has row (1,A) and target table TCTA has row (2,1,C1).
   g. The source and target tables are now out of sync.
2. Scenario 2
   a. Parent row (2,B) and corresponding child row (2,2,C2) exist in tables SPTA and SCTA on the source, respectively. Identical rows exist in tables TPTA and TCTA on the target.
   b. The existing child row (2,2,C2) in the source table SCTA is deleted first, followed by a delete of the parent row (2,B). These changes are not propagated to the target when failover occurs.

c. After failover, the non-key column Col2 of the existing child row (2,2,C2) is updated to a value of Z2. This change is not propagated to the source since the source is unavailable.

d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

e. Propagation from target to source:

   i. The update operation from the target table TCTA does not find an existing row in the source table SCTA and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is force, the update operation is converted to an insert operation and an attempt is made to insert the row into the source table SCTA. The IS_APPLIED value is set to 'Y'.

      However, this insert operation fails because the corresponding parent row is missing. An SQL error exception is generated with SQLCODE -530 indicating a missing parent row. Since the SQLSTATE 23503 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

   ii. At this point, the source tables SPTA and SCTA are empty.

f. Propagation from source to target:

   i. The delete operation of the child row (2,2,C2) from the source table SCTA encounters a row with a matching key in the target table TCTA, but finds a mismatch on Col2. The before value of the changed column Col2 was C2 in the source table SCTA, but the before value of Col2 was Z2 in the target table TCTA. An exception is generated with an SQLCODE 100 and a reason of CHECKFAILED. Since the conflict action is ignore, it ignores the delete operation propagated from the source and leaves the existing row in place. It records this ignore action by setting the IS_APPLIED value to 'N'.

   ii. The delete operation of the parent row (2,B) from the source table SPTA finds a matching key and matching changed non-key columns row in the target table TPTA and attempts to delete it. However, since the delete rule is RESTRICT and the parent row (2,B) in the target table TPTA has a child row (2,2,Z2) in the target table TCTA, the delete operation fails. An SQL error exception is generated with an SQLCODE -532 indicating a referential constraint violation. Since the SQLSTATE 23001 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

   iii. At this point, the target table TPTA has row (2,B) and target table TCTA has row (2,2,Z2).

g. The source and target tables are now out of sync.

3. Scenario 3

   a. Parent row (3,C) and corresponding child row (3,3,C3) exist in tables SPTA and SCTA on the source, respectively. Identical rows exist in tables TPTA and TCTA on the target.

   b. The existing child row (3,3,C3) in the source table SCTA is deleted first, followed by a delete of the parent row (3,C). These changes are not propagated to the target when failover occurs.

   c. After failover, a child row (31,3,C9) is inserted into the target table TCTA. This change is not propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

e. Propagation from target to source:

   i. The insert operation from the target table TCTA does not find an existing row in the source table SCTA, and attempts to insert it.

      However, this insert operation fails because the corresponding parent row is missing. An SQL error exception is generated with SQLCODE -530 indicating a missing parent row. Since the SQLSTATE 23503 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

   ii. At this point, the source tables SPTA and SCTA are empty.

f. Propagation from source to target:

   i. The delete operation of the child row (3,3,C3) from the source table SCTA encounters a row with a matching key and matching non-key columns in the target table TCTA and deletes it. No exception is generated.

   ii. The delete operation of the parent row (3,C) from the source table SPTA finds a matching key and matching changed non-key columns row in the target table TPTA and attempts to delete it. However, since the delete rule is RESTRICT and the parent row (3,C) in the target table TPTA has a child row (31,3,C9) in the target table TCTA, the delete operation fails. An SQL error exception is generated with an SQLCODE -532 indicating a referential constraint violation. Since the SQLSTATE 23001 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

   iii. At this point, the target table TPTA has row (3,C) and target table TCTA has row (31,3,C9).

g. The source and target tables are now out of sync.

4. Scenario 4

   a. Parent row (4,D) exists in source table SPTA and target table TPTA.

   b. The key of the parent row (4,D) in the source table SCTA is updated to a value of 99. This change is not propagated to the target when failover occurs.

   c. After failover, a child row (4,4,C4) is inserted into the target table TCTA. This change is not propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   e. Propagation from target to source:

      i. The insert operation from the target table TCTA does not find an existing row in the source table SCTA, and attempts to insert it.

         However, this insert operation fails because the corresponding parent row is missing. An SQL error exception is generated with SQLCODE -530 indicating a missing parent row. Since the SQLSTATE 23503 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

      ii. At this point, the source table SPTA has row (99,D), while the source table SCTA is empty.

   f. Propagation from source to target:

      i. The update operation of the key of the parent row (99,D) from the source table SPTA encounters a row with a matching key and matching non-key columns in the target table TCTA and attempts to make the change.

However, since the update rule is RESTRICT and the parent row (4,D) in the target table TPTA has a child row (4,4,C4) in the target table TCTA, the update operation fails. An SQL error exception is generated with an SQLCODE -531 indicating a referential constraint violation. Since the SQLSTATE 23504 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

ii. At this point, the target table TPTA has row (4,D) and target table TCTA has row (4,4,C4).

g. The source and target tables are now out of sync.

5. Scenario 5

a. Parent row (5,A) and corresponding child row (5,5,C5) exist in tables SPTA and SCTA on the source, respectively. Identical rows exist in tables TPTA and TCTA on the target.

b. The key of the existing child row (5,5,C5) is updated to a value of 8. This change is not propagated to the target when failover occurs.

c. After failover, the existing child row (5,5,C5) in the target table TCTA is deleted first, followed by a delete of the parent row (5,A). These changes are not propagated to the source since the source is unavailable.

d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

e. Propagation from target to source:

i. The delete operation of the child row (5,5,C5) from the target table TCTA does not find an existing row in the source table SCTA and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is force, and the row does not exist, the operation is assumed to have completed successfully, and the IS_APPLIED value is set to 'Y'.

ii. The delete operation of the parent row (5,A) from the target table TPTA finds a matching key and matching changed non-key columns row in the source table SPTA and attempts to delete it. However, since the delete rule is RESTRICT and the parent row (5,A) in the source table SPTA has a child row (8,5,C5) in the source table SCTA, the delete operation fails. An SQL error exception is generated with an SQLCODE -532 indicating a referential constraint violation. Since the SQLSTATE 23001 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

iii. At this point, the source table SPTA has row (5,A) and source table SCTA has row (8,5,C5).

f. Propagation from source to target:

i. The update operation from the source table SCTA does not find an existing row in the target table TCTA and records an exception with an SQLCODE 100 indicating a missing row. Since the conflict action is ignore, the IS_APPLIED value is set to 'N'.

ii. At this point, the target tables TPTA and TCTA are empty.

g. The source and target tables are now out of sync.

6. Scenario 6

a. Parent row (6,C) and corresponding child row (6,6,C6) exist in tables SPTA and SCTA on the source, respectively. Identical rows exist in tables TPTA and TCTA on the target.

b. A child row (61,6,C9) is inserted into the source table SCTA. This change is not propagated to the target when failover occurs.

c. After failover, the existing child row (6,6,C6) in the target table TCTA is deleted first, followed by a delete of its parent row (6,C). These changes are not propagated to the target when failover occurs.

d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

e. Propagation from target to source:

   i. The delete operation of the child row (6,6,C6) from the target table TCTA finds a matching key and matching changed non-key columns row in the source table SPTA and deletes it. No exception is generated.

   ii. The delete operation of the parent row (6,C) from the target table TPTA finds a matching key and matching changed non-key columns row in the source table SPTA and attempts to delete it. However, since the delete rule is RESTRICT and the parent row (6,C) in the source table SPTA has a child row (61,6,C9) in the source table SCTA, the delete operation fails. An SQL error exception is generated with an SQLCODE -532 indicating a referential constraint violation. Since the SQLSTATE 23001 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

   iii. At this point, the source table SPTA has row (6,C) and the source table SCTA has row (61,6,C9).

f. Propagation from source to target:

   i. The insert operation of the child row (61,6,C9) from the source table SCTA does not find an existing row in the target table TCTA and attempts to insert it.

      However, this insert operation fails because the corresponding parent row is missing. An SQL error exception is generated with SQLCODE -530 indicating a missing parent row. Since the SQLSTATE 23503 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to 'N'.

   ii. At this point, the target tables TPTA and TCTA are empty.

g. The source and target tables are now out of sync.

7. Scenario 7

   a. Parent row (7,E) exists in source table SPTA and target table TPTA.

   b. A child row (7,7,C7) in inserted into the source table SCTA. This change is not propagated to the target when failover occurs.

   c. After failover, the key of the parent row (7,E) in the target table TPTA is updated to a value of 97. This change is not propagated to the source since the source is unavailable.

   d. When the source becomes available and switchback is initiated, these unpropagated changes are propagated to the appropriate servers.

   e. Propagation from target to source:

      i. The update operation of the key of the parent row (97,E) from the target table TPTA encounters a row with a matching key and matching non-key columns in the source table SPTA and attempts to make the change.

         However, since the update rule is RESTRICT and the parent row (7,E) in the source table SPTA has a child row (7,7,C7) in the source table SCTA, the update operation fails. An SQL error exception is generated with an SQLCODE -531 indicating a referential constraint violation. Since the SQLSTATE 23504 is included in the

OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to ‘N’.

ii. At this point, the source table SPTA has row (7,E) and source table SCTA has row (7,7,C7).

f. Propagation from source to target:

i. The insert operation of the child row (7,7,C7) from the source table SCTA does not find an existing row in the target table TCTA and attempts to insert it.

However, this insert operation fails because the corresponding parent row is missing. An SQL error exception is generated with SQLCODE -530 indicating a missing parent row. Since the SQLSTATE 23503 is included in the OKSQLSTATES column of the IBMQREP_TARGETS table, the Reason field has OKSQLSTATE, and IS_APPLIED is set to ‘N’.

ii. At this point, the target table TPTA has row (97,E) while the target table TCTA is empty.

g. The source and target tables are now out of sync.

C

WebSphere shared queues and shared disk overview

In this appendix we provide an overview of WebSphere shared queues and shared disk implementations, and summarize the considerations for choosing one or the other in a high availability environment.

The topics covered include:

- WebSphere shared queues overview
- WebSphere shared disk overview
- Shared queues versus shared disk implementation

C.1 WebSphere shared queues overview

Shared queues are local queues that can be accessed by one or more queue managers in a Parallel Sysplex®. Each queue manager that should have access to these shared queues must be a member of a queue-sharing group. Using shared queues with your applications, you are able to exploit advantages of the Parallel Sysplex, such as high availability, workload balancing, and reduction of administrative and operational work.

In order to use shared queues, a queue manager must become a member of a queue-sharing group. A queue-sharing group is a collection of queue managers in a Parallel Sysplex that have access to the same set of shared queues and shared definitions.

Figure C-1 shows a setup of a queue-sharing group QSG1 with two queue managers, WMQA and WMQB.

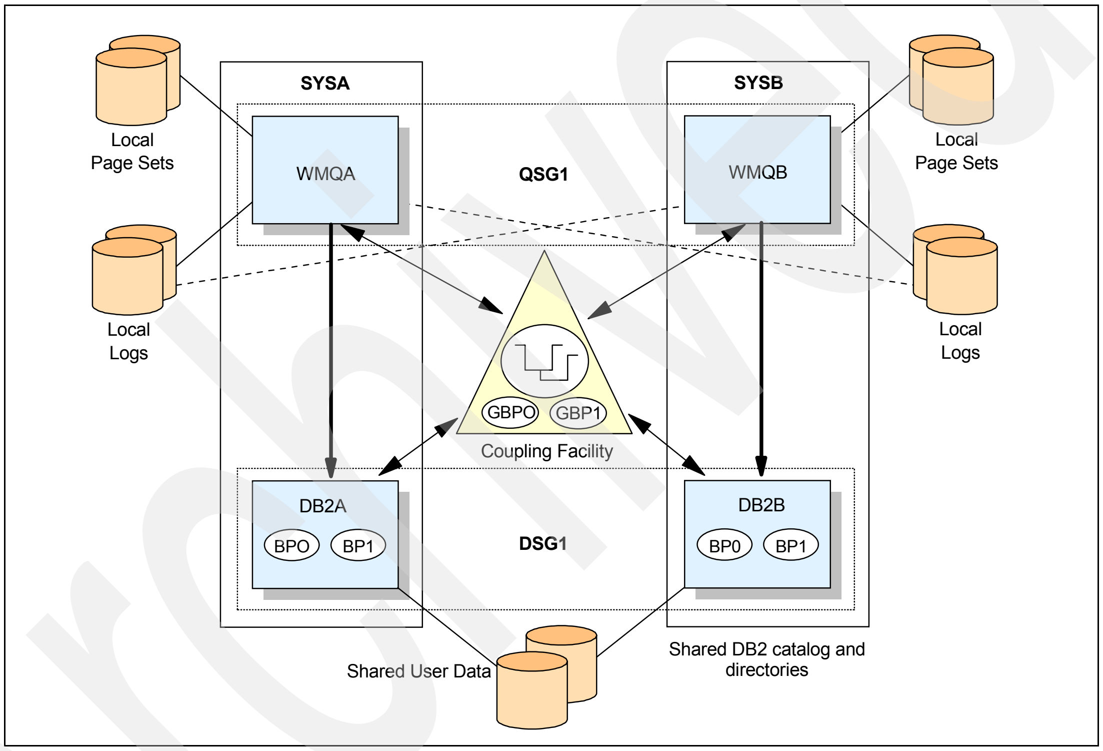


Figure C-1 Queue-sharing group in a Parallel Sysplex

Both queue managers still have their own local bootstrap data sets, log data sets, and page sets, but in order to allow a queue-sharing group wide recovery, all queue managers must have access to their own log data sets as well as the log data sets of all other queue managers in the queue-sharing group.

The shared WebSphere MQ object definitions are stored in DB2 tables, and messages belonging to a shared queues reside in one or more coupling facility list structures.

Each queue manager connects to a DB2 subsystem that has to be a member of a DB2 data sharing group. In our example, the DSG1 is the data sharing group with two members, DB2A and DB2B.

Multiple queue managers that are members of the same queue-sharing group (QSG) have access to shared queues contained within that QSG. Each shared queue is available to all queue managers that are members of the QSG as if they were local to the queue manager.

This means that one application connected to a queue manager can put a message to a shared queue, and then a second application connected to a second queue manager within the QSG can get that message from the same shared queue. Without the functionality of shared queues, this message would need to be transferred to a queue hosted on the second queue manager by a distributed or cluster message channel before the second application could get the message.

Another significant benefit of a QSG is that if one queue manager within a QSG fails, the other queue managers within the QSG can continue to process data from shared queues contained within that QSG. *This is the feature exploited in this scenario involving unidirectional Q replication*.

QSGs build on the functionality provided by connecting z/OS images together in a sysplex. All queue managers that are members of a QSG must be hosted on z/OS images within the same sysplex.

The WebSphere MQ for z/OS product uses the coupling facility (CF)[1] in combination with the facilities provided by the IBM DB2 database product. Therefore, each queue manager within a QSG must have access to DB2. The DB2 instances that the queue managers within a QSG access must be within the same data sharing group. A data sharing group is a DB2 feature that enables multiple DB2 instances to share a repository of information.

WebSphere MQ for z/OS uses the CF and DB2 to share the definition of the queue, and the messages contained within that queue, between the queue managers that are members of the QSG. When defined on one queue manager within the QSG, all queue managers within the QSG can access a shared queue. Each QSG has a name.

Note: With WebSphere MQ for z/OS Version 6.0 the maximum length of a message that can be put onto a shared queue has been increased from 63 KB to 100 MB. When a message larger than 63 KB is put on a shared queue, a placeholder is stored in the CF (4 KB) and the message data is stored using DB2.

Prior to setting up a WebSphere MQ queue-sharing group, you have to ensure that all the queue managers that you want to be members of the queue-sharing group are running in the same Parallel Sysplex. The maximum number of queue managers that you can connect to a queue-sharing group is 32.

The various resources used by a queue-sharing group include system resources and DB2 resources, as follows.

System resources

The system resources used by a queue-sharing group are the coupling facility and resource recovery services, as follows:

- Coupling facility

  WebSphere MQ needs a coupling facility with at least CFLEVEL=9 to process shared messages. This is because CFLEVEL=9 provides XES coupling facility list structure architecture extensions to aid in data management and manipulation in support of WebSphere MQ shared queues.

  The coupling facility storage requirements for your queue-sharing group are mainly dependent on:

  – The number of shared queues you want to define

[1] A sysplex provides a coupling facility that allows multiple z/OS images within a sysplex to share data.

- The number of messages you expect to be in a shared queue at any time

  WebSphere MQ uses coupling facility list structures for administrative purposes and for shared queues. You have to define at least two coupling facility structures in your coupling facility resource management (CFRM) policy to use shared queues:

  a. One administration structure:

     The administration structure does not contain any user data. WebSphere MQ uses the administration structure to coordinate internal activity across the queue-sharing group. This allows all queue managers in the queue-sharing group to exchange and control information.

  b. Application structures:

     Application structures hold shared messages, which are put to a shared queue.

     Depending on your application requirements, you can define one or more application structures, up to a maximum of 63, because the limit for the number of structures a queue manager can connect to is 64, and one is needed for the administration structure. All messages associated with a single shared queue are in a single application structure. A single application structure can contain messages for up to 512 shared queues.

- Resource Recovery Services (RRS)

  In order to communicate with a DB2 data sharing group, you also have to ensure that RRS runs on all images on which you want to implement queue-sharing group queue managers.

DB2 resources

To share data in a queue-sharing group, WebSphere MQ uses the coupling facility list structures for the shared messages, a set of DB2 tables for the object definitions, and status information. To ensure that all queue managers in a queue-sharing group share the same set of tables, you need to set up a DB2 data sharing group.

There are three types of tables, which are put into different table spaces:

- Administration tables and all object tables apart from OBJ_B_NAMELIST and OBJ_B_CHANNEL.
- OBJ_B_NAMELIST and OBJ_B_CHANNEL tables. The table space for this table must be associated with a DB2 32 KB buffer pool.
- Channel initiator tables.

Refer to *WebSphere MQ for z/OS Concepts and Planning Guide*, GC34-6051, for a description of all shipped tables and the storage requirements of these tables.

With the exception of the queue-sharing group and queue manager entries, WebSphere MQ adds, updates, and deletes all DB2 entries automatically, when you change your shared queue or group object definitions. In order to manage queue-sharing group and queue manager definitions, WebSphere MQ provides a utility, CSQ5PQSG.

WebSphere MQ queue managers in the queue-sharing group connect to a local DB2 of the data sharing group. A queue manager needs both:

- The DB2 data sharing group name
- The name of the local DB2 subsystem or group attachment (which is defined through the subsystem entry of the DB2 subsystem) to which the queue manager is to connect

You have to provide both parameters in the parameter module CSQ6SYSP, and you have to create the queue-sharing group and queue manager entries in the DB2 tables.

For further details on installing and configuring shared queues, refer to the IBM Redbook *WebSphere MQ in a z/OS Parallel Sysplex Environment*, SG24-6864.

C.2 WebSphere shared disk overview

With a WebSphere MQ shared disk configuration, a cluster of servers cooperate to share access to disks to provide high availability of services running within the cluster. The most basic configuration has one node performing work while the other node acts only as standby (it does not perform work and is referred to as idle). This configuration is sometimes called cold standby.

Figure C-2 shows a typical WebSphere MQ shared disk configuration where WebSphere MQ queue manager QM1 is defined on both nodes, and share access to local and remote queues on shared disk. When both queue managers share the same name, only one of them can be started at any one time. For example, Node A has the queue manager QM1 that has been started performing the work, while Node B is in standby mode with the queue manager QM1 not being started[2].

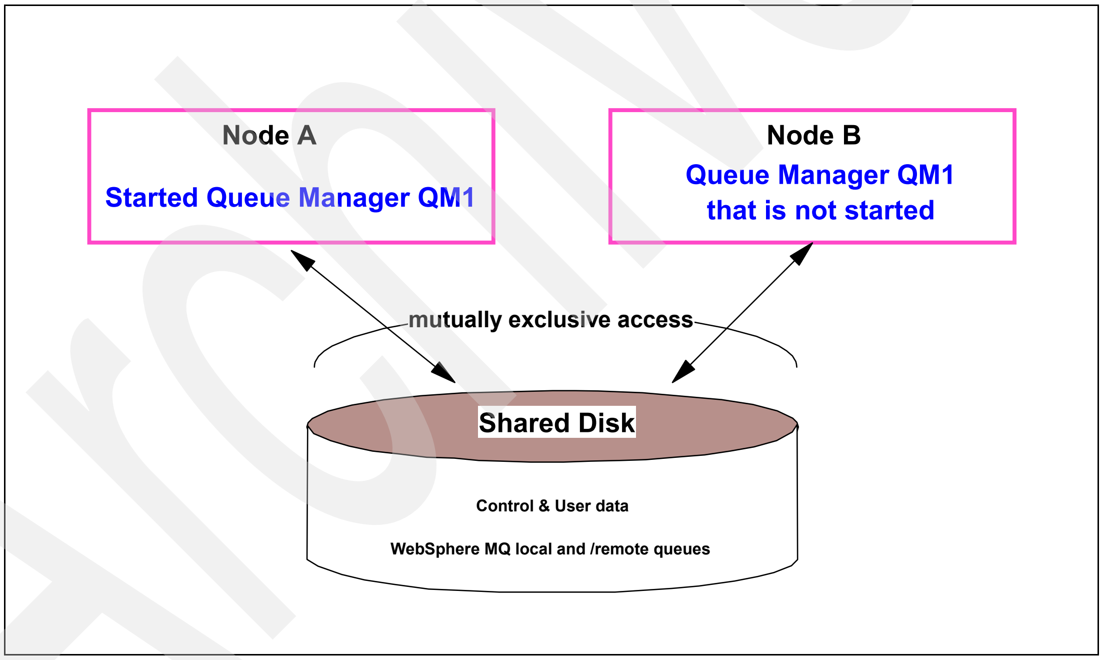

Node A
Started Queue Manager QM1
Node B
Queue Manager QM1
that is not started
mutually exclusive access
Shared Disk
Control & User data
WebSphere MQ local and /remote queues

Figure C-2 Typical WebSphere MQ shared disk cold standby configuration

C.3 Shared queues versus shared disk implementation

Nodes communicate with WebSphere MQ queues through the coupling facility with the shared queues implementation, as compared to the shared disk implementation, which uses disk storage. In WebSphere MQ Version 6, the maximum length of a message that can be put

[2] If you attempt to start QM1 on Node B while it is started on Node A, an error message will result and the startup of QM1 on Node B will fail.

in a shared queue has increased from 63 KB to 100 MB[3]. There are no such limitations with the shared disk implementation.

A potential disadvantage of shared queues using the coupling facility is that if messages build in the coupling facility, it will have a negative impact on the performance of other subsystems such as DB2 using the coupling facility. No such disadvantage exists with the shared disk implementation. However, if the messages are consumed quickly then the shared queues implementation performance is superior to that of the shared disk implementation.

A potential disadvantage of the shared disk implementation is that the queue manager has to be started on the other node before Q replication can be started on that node. No such requirement exists with the shared queues implementation.

[3] As mentioned earlier, when a message is greater than 63KB, a placeholder (4KB) is stored in the coupling facility and the message is stored in DB2.

D

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG247215

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG247215.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
BIDI.zip	Code and scripts to set up the environment and perform the steps for controlled failover; switchback from controlled failover; uncontrolled failover; and switchback from uncontrolled failover options A, B, C, and D.
SHQUEUE.zip	Code and scripts to set up the WebSphere MQ shared queues high availability unidirectional Q replication environment and perform the steps for failover with no inflight automatic load in progress, and failover with inflight automatic load in progress.
SHDISK.zip	Code and scripts to set up the WebSphere MQ shared disk high availability unidirectional Q replication environment and perform the steps for failover with no inflight automatic load in progress, and failover with inflight automatic load in progress.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space	2 MB
Operating System	Windows

How to use the Web material

Create a subdirectory SG247215 on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

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

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 260. Note that some of the documents referenced here may be available in softcopy only.

- *MQSeries Primer,* REDP-0021-00
- *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487
- *WebSphere MQ in a z/OS Parallel Sysplex Environment*, SG24-6864
- *WebSphere MQ V6 Fundamentals*, SG24-7128
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance,* SG24-6517

Other publications

These publications are also relevant as further information sources:

- *IBM WebSphere for Replication Server, Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568-00
- *IBM WebSphere for Replication Server, ASNCLP Program Reference for Replication and Event Publishing Version8.2*, SC18-9410-00
- *WebSphere MQ for AIX, Quick Beginnings Version5.3*, GC34-6079-01
- *WebSphere MQ, System Administration Guide*, SC34-6069-01
- *WebSphere MQ, Script (MQSC) Command Reference*, SC34-6055-01
- *WebSphere MQ, Security Version5.3*, SC34-6079-01
- *WebSphere MQ, Intercommunication*, SC34-6059-01

Online resources

These Web sites and URLs are also relevant as further information sources:

- Roadmap for Q Replication

  `http://www-128.ibm.com/developerworks/db2/roadmaps/qrepl-roadmap-v8.2.html`
- DB2 Information Center

  `http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp`
- DB2 UDB Version 8 manuals site

  `http://www-306.ibm.com/software/data/db2/udb/support/manualsv8.html`

- WebSphere MQ sites

  http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/
  http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/csqzak05/csqzak055e.htm

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

WebSphere Replication Server for z/OS Using Q Replication: High Availability Scenarios for the z/OS Platform

WebSphere Replication Server for z/OS Using Q Replication

High Availability Scenarios for the z/OS Platform

WebSphere Replication Server for z/OS Q replication overview

Bidirectional Q replication failover/switchback scenarios

WebSphere MQ shared disk/shared queue scenarios

This IBM Redbook provides detailed instructions and scripts for managing failover and switchback in a WebSphere Replication Server for z/OS bidirectional Q replication environment for the z/OS platform. A typical business scenario is used to showcase the bidirectional failover/switchback implementation. This book also includes a WebSphere MQ shared disk and WebSphere MQ shared queue high availability scenario for the source system in a Q replication environment involving unidirectional replication. Key considerations in designing and implementing such environments are discussed.

This book is aimed at an audience of IT architects and database administrators (DBAs) responsible for developing high availability solutions on the z/OS platform.

SG24-7215-00 ISBN 0738497126