

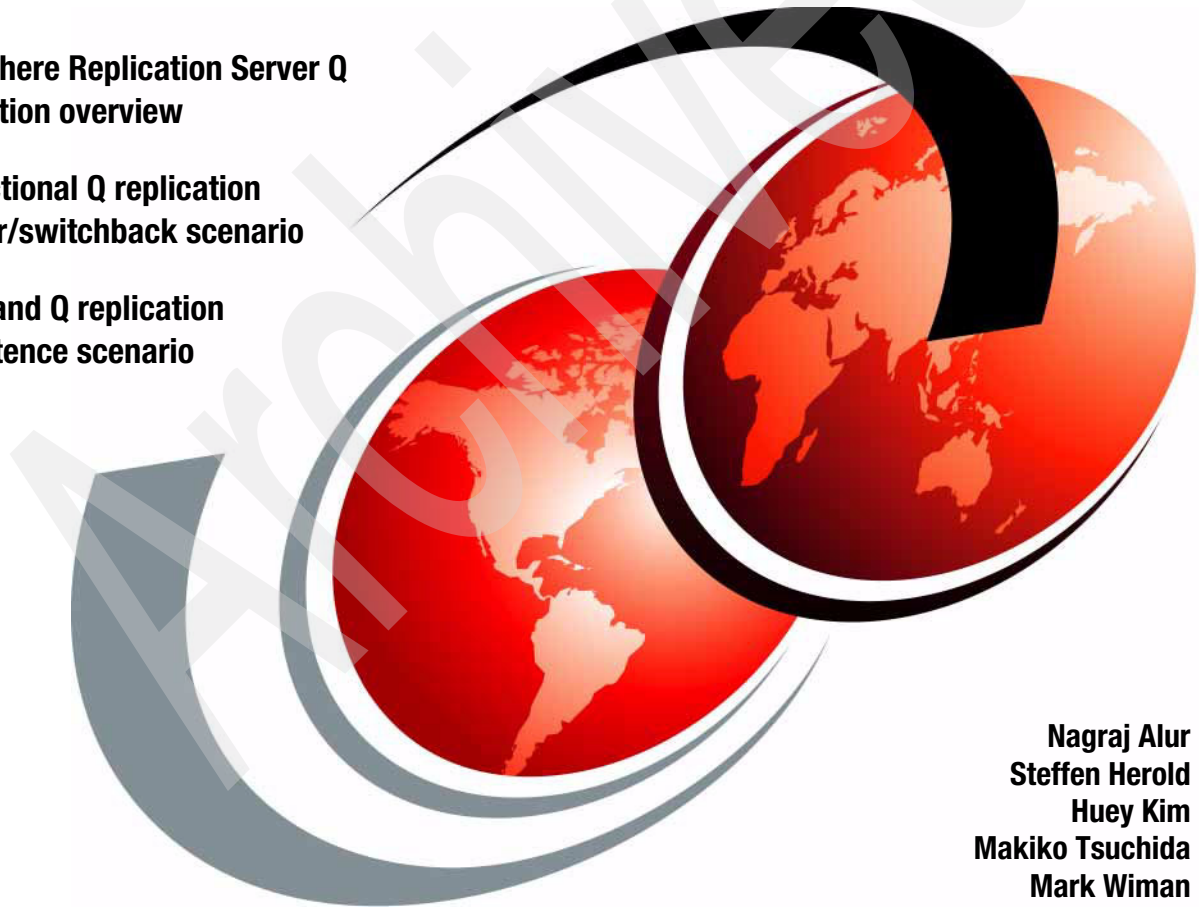
WebSphere Replication Server Using Q Replication

High Availability Scenarios for the AIX Platform

**WebSphere Replication Server Q
replication overview**

**Bidirectional Q replication
failover/switchback scenario**

**HADR and Q replication
coexistence scenario**



**Nagraj Alur
Steffen Herold
Huey Kim
Makiko Tsuchida
Mark Wiman**



International Technical Support Organization

**WebSphere Replication Server Using Q Replication
High Availability Scenarios for the AIX Platform**

June 2006

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page xvii.

First Edition (June 2006)

This edition applies to Version 9, Release 1 Modification 0 of WebSphere Q Replication Server (expected product number 5724-N98).

© Copyright International Business Machines Corporation 2006. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

| | |
|---|-------|
| Figures | vii |
| Tables | ix |
| Examples | xi |
| Notices | xvii |
| Trademarks | xviii |
| Preface | xix |
| The team that wrote this redbook | xx |
| Become a published author | xxii |
| Comments welcome | xxii |
| Chapter 1. High availability concepts | 1 |
| 1.1 Introduction | 2 |
| 1.2 Levels of availability | 3 |
| Chapter 2. WebSphere Replication Server Q replication overview | 9 |
| 2.1 Q replication overview | 10 |
| 2.1.1 Unidirectional replication | 12 |
| 2.1.2 Bidirectional replication | 12 |
| 2.1.3 Peer-to-peer replication | 13 |
| 2.2 Q replication processing flow | 13 |
| 2.2.1 Initial synchronization of the data at the source and target | 13 |
| 2.2.2 Ongoing replication after the initial synchronization | 19 |
| 2.3 Choosing a particular Q replication topology | 28 |
| 2.4 Latency considerations | 29 |
| 2.4.1 Q Capture latency | 30 |
| 2.4.2 Q Capture transaction latency | 31 |
| 2.4.3 Queue latency | 31 |
| 2.4.4 Q Apply latency | 32 |
| 2.4.5 End-to-end latency | 32 |
| Chapter 3. Failover and switchback scenarios | 33 |
| 3.1 Introduction | 34 |
| 3.2 Business requirement | 34 |
| 3.3 Rationale for the bidirectional solution | 35 |
| 3.4 Environment configuration | 36 |
| 3.5 Failover and switchback considerations | 41 |

| | | |
|-------------------|---|------------|
| 3.5.1 | Failover processing considerations | 41 |
| 3.5.2 | Switchback processing considerations | 44 |
| 3.6 | Controlled failover and switchback | 49 |
| 3.6.1 | Controlled failover (CF) | 50 |
| 3.6.2 | Switchback after CF with no data loss | 56 |
| 3.7 | Uncontrolled failover (UF) and switchback | 70 |
| 3.7.1 | Uncontrolled failover (UF) | 72 |
| 3.7.2 | Option A: Automatically resync primary and secondary servers | 76 |
| 3.7.3 | Option B: Resync primary and reinitialize secondary | 84 |
| 3.7.4 | Option C: Resync secondary and reinitialize primary | 102 |
| 3.7.5 | Option D: Discard primary changes and reinitialize primary | 109 |
| Chapter 4. | HADR and Q replication coexistence scenario | 117 |
| 4.1 | Introduction | 118 |
| 4.2 | Business requirement | 118 |
| 4.3 | Rationale for the unidirectional solution | 119 |
| 4.4 | Environment configuration | 119 |
| 4.4.1 | HA primary environment using HADR and DB2 Client Reroute | 120 |
| 4.4.2 | Unidirectional replication environment | 143 |
| 4.5 | Set up of the environment | 145 |
| 4.5.1 | STEP SETHADR1: Create the test tables | 146 |
| 4.5.2 | STEP SETHADR2: Set up appropriate WebSphere MQ | 148 |
| 4.5.3 | STEP SETHADR3: Set up unidirectional replication | 150 |
| 4.6 | Planned takeover | 156 |
| 4.6.1 | STEP TESTPTKO1: Start Q Apply on both secondary servers | 157 |
| 4.6.2 | STEP TESTPTKO2: Start Q Capture on HADR primary | 159 |
| 4.6.3 | STEP TESTPTKO3: Connect to HADR primary and view state | 160 |
| 4.6.4 | STEP TESTPTKO4: Run workload on client | 161 |
| 4.6.5 | STEP TESTPTKO5: Stop Q Capture on HADR primary and verify the HADR status | 162 |
| 4.6.6 | STEP TESTPTKO6: Takeover HADR on standby and verify HADR status | 164 |
| 4.6.7 | STEP TESTPTKO7: Start the Q Capture on the new HADR primary (twice) | 171 |
| 4.6.8 | STEP TESTPTKO8: Run workload on client and verify state | 174 |
| 4.6.9 | STEP TESTPTKO9: Verify Q replication up and running | 178 |
| 4.6.10 | STEP TESTPTKO10: Verify all workload changes applied to target tables | 182 |
| 4.7 | Unplanned takeover | 185 |
| 4.7.1 | STEP TESTUTKO1: Start Q Apply on both secondary servers | 186 |
| 4.7.2 | STEP TESTUTKO2: Start Q Capture on HADR primary | 186 |
| 4.7.3 | STEP TESTUTKO3: Connect to HADR primary and view state | 187 |
| 4.7.4 | STEP TESTUTKO4: Run workload on client | 187 |

| | | |
|--|--|------------|
| 4.7.5 | STEP TESTUTKO5: Stop Q Capture on HADR primary and verify the HADR status | 187 |
| 4.7.6 | STEP TESTUTKO6: Run workload on client to build unpropagated changes on HADR primary | 187 |
| 4.7.7 | STEP TESTUTKO7: Stop DB2 force on HADR primary to simulate failure of HADR primary | 188 |
| 4.7.8 | STEP TESTUTKO8: Takeover HADR on standby by FORCE and verify HADR status | 188 |
| 4.7.9 | STEP TESTUTKO9: Start the Q Capture on the HADR new primary (twice) | 193 |
| 4.7.10 | STEP TESTUTKO10: Run workload on client and verify state . . . | 193 |
| 4.7.11 | STEP TESTUTKO11: Verify Q replication up and running | 195 |
| 4.7.12 | STEP TESTUTKO12: Verify all workload changes applied to target tables | 195 |
| 4.8 | Failback | 198 |
| 4.8.1 | STEP TESTREVERT1: Start DB2 on HADR old primary | 199 |
| 4.8.2 | STEP TESTREVERT2: Start HADR on old primary as standby site and verify HADR status | 199 |
| 4.8.3 | STEP TESTREVERT3: Stop Q Capture on HADR current primary | 206 |
| 4.8.4 | STEP TESTREVERT4: Failback and verify HADR status | 207 |
| 4.8.5 | STEP TESTREVERT5: Start Q Capture on HADR primary (twice) | 214 |
| 4.8.6 | STEP TESTREVERT6: Run workload on client and verify state . . | 214 |
| 4.8.7 | STEP TESTREVERT7: Verify Q replication up and running | 218 |
| 4.8.8 | STEP TESTREVERT8: Verify all workload changes applied to target tables | 218 |
| Appendix A. Summary of code and scripts used in the scenarios | | 223 |
| A.1 | Summary | 224 |
| Appendix B. Exception processing in a bidirectional Q replication environment | | 225 |
| B.1 | Exceptions overview | 226 |
| B.1.1 | Conflict considerations | 230 |
| B.1.2 | SQL errors considerations | 236 |
| B.1.3 | Viewing exceptions in the IBMQREP_EXCEPTIONS table | 239 |
| B.1.4 | Considerations for conflict detection in HA scenarios | 246 |
| B.2 | Sample exceptions in a bidirectional scenario | 247 |
| B.2.1 | Exceptions with inserts | 254 |
| B.2.2 | Exceptions with deletes | 263 |
| B.2.3 | Exceptions with updates | 269 |
| B.3 | Data inconsistencies in HA bidirectional scenario | 276 |
| B.3.1 | Non-referential integrity related data inconsistency scenarios | 278 |
| B.3.2 | Referential integrity related data inconsistency scenarios | 291 |

| | |
|--|-----|
| Appendix C. Overview of HADR and DB2 Client Reroute | 301 |
| C.1 HADR | 302 |
| C.2 DB2 Client Reroute | 305 |
| Appendix D. Additional material | 307 |
| Locating the Web material | 307 |
| Using the Web material | 307 |
| System requirements for downloading the Web material | 308 |
| How to use the Web material | 308 |
| Related publications | 309 |
| IBM Redbooks | 309 |
| Other publications | 309 |
| Online resources | 310 |
| How to get IBM Redbooks | 310 |
| Help from IBM | 310 |
| Index | 311 |

Figures

| | |
|---|-----|
| 1-1 Levels of availability and costs | 4 |
| 1-2 Availability chains | 5 |
| 2-1 A simple Q replication configuration | 10 |
| 2-2 Specifying how target table is to be loaded for given Q subscription | 14 |
| 2-3 Activating Q subscription via the Replication Center | 15 |
| 2-4 Manual load processing flow in unidirectional Q replication | 16 |
| 2-5 Ongoing replication flow | 19 |
| 2-6 Latency statistics collected | 29 |
| 2-7 Computing latency statistics | 30 |
| 3-1 Bidirectional replication topology configuration | 36 |
| 3-2 Tables used in the bidirectional replication scenario | 37 |
| 3-3 Bidirectional replication topology objects overview | 40 |
| 3-4 Overview of set up steps for the bidirectional replication environment | 41 |
| 3-5 Data loss and potential sources of conflicts | 42 |
| 3-6 Bidirectional replication high availability failover and switchback options | 49 |
| 3-7 Overview of controlled failover (CF) steps | 51 |
| 3-8 Overview of switchback after controlled failover (CF) steps | 57 |
| 3-9 Q Capture latency and end-to-end latency using Replication Center 1/4 | 62 |
| 3-10 Q Capture latency and end-to-end latency using Replication Center 2/4 | 63 |
| 3-11 Q Capture latency and end-to-end latency using Replication Center 3/4 | 63 |
| 3-12 Q Capture latency and end-to-end latency using Replication Center 4/4 | 64 |
| 3-13 Viewing exceptions through Replication Center 1/4 | 68 |
| 3-14 Viewing exceptions through Replication Center 2/4 | 69 |
| 3-15 Viewing exceptions through Replication Center 3/4 | 69 |
| 3-16 Viewing exceptions through Replication Center 4/4 | 70 |
| 3-17 Overview of uncontrolled failover (UF) steps | 73 |
| 3-18 Overview of Option A steps | 77 |
| 3-19 Overview of Option B steps | 86 |
| 3-20 Overview of Option C steps | 104 |
| 3-21 Overview of Option D steps | 111 |
| 4-1 Unidirectional replication topology configuration | 120 |
| 4-2 HADR and DB2 Client Reroute synergy | 121 |
| 4-3 Overview of steps to set up HADR | 123 |
| 4-4 Overview of steps to set up DB2 Client Reroute | 135 |
| 4-5 Tables used in the unidirectional replication scenario | 145 |
| 4-6 Overview of steps to set up the unidirectional scenario | 145 |
| 4-7 Overview of steps to set up the unidirectional replication environment | 150 |
| 4-8 Overview of planned takeover steps | 157 |

| | |
|--|-----|
| 4-9 Overview of unplanned takeover steps | 186 |
| 4-10 Overview of failback | 199 |
| B-1 Conflict rule for a subscription in bidirectional Q replication | 230 |
| B-2 Conflict action for a subscription in bidirectional Q replication. | 233 |
| B-3 Specifying OKSQLSTATES for subscription in Replication Center | 237 |
| B-4 Error action for a subscription - both bidirectional and peer-to-peer replica- tion | 238 |
| B-5 Using Replication Center to view exceptions 1/5. | 241 |
| B-6 Using Replication Center to view exceptions 2/5. | 242 |
| B-7 Using Replication Center to view exceptions 3/5. | 242 |
| B-8 Using Replication Center to view exceptions 4/5. | 243 |
| B-9 Using Replication Center to view exceptions 5/5. | 244 |
| B-10 Using Live Monitor to view exceptions 1/2. | 245 |
| B-11 Using Live Monitor to view exceptions 2/2. | 246 |
| B-12 Bidirectional Q replication HA environment - normal operations | 248 |
| B-13 Bidirectional Q replication HA environment - failover operations. | 249 |
| B-14 Bidirectional Q replication HA environment - switchback operations. | 250 |
| B-15 Propagated INSERTs processing | 251 |
| B-16 Propagated DELETes processing | 252 |
| B-17 Propagated UPDATES processing (key and non key columns, or only the non key columns) | 253 |
| B-18 Tables used to demonstrate the data inconsistency scenarios | 278 |
| C-1 HADR overview | 302 |
| C-2 Different synchronization modes of HADR | 303 |
| C-3 DB2 Client Reroute | 305 |

Tables

| | | |
|-----|---|-----|
| 1-1 | Availability metric - nine rule | 6 |
| 2-1 | Choosing a particular Q replication topology | 28 |
| 3-1 | Criteria for selecting the appropriate switchback procedure | 46 |
| B-1 | IBMQREP_EXCEPTIONS table | 226 |
| B-2 | Non-referential integrity related data inconsistent scenarios | 279 |
| B-3 | Referential integrity related data inconsistent scenarios | 292 |

Archived

Examples

| | |
|---|----|
| 2-1 Insert a CAPSTART signal in the IBMQREP_SIGNAL table | 15 |
| 2-2 Inserting a row in the IBMQREP_SIGNAL table | 17 |
| 3-1 DDL of tables used in the bidirectional replication scenario. | 37 |
| 3-2 Autostop Q Capture on the primary server DENMARK | 52 |
| 3-3 Autostop Q Capture log contents on the primary server DENMARK. | 52 |
| 3-4 Stop Q Capture on the secondary server JAMAICA | 53 |
| 3-5 Stop Q Capture log contents on the secondary server JAMAICA | 53 |
| 3-6 Stop Q Apply on the primary server DENMARK | 54 |
| 3-7 Stop Q Apply log contents on the primary server DENMARK | 54 |
| 3-8 Verify XMITQ on the primary server DENMARK is fully consumed. | 54 |
| 3-9 Take down the primary server DENMARK. | 55 |
| 3-10 Autostop Q Apply on the secondary server JAMAICA. | 55 |
| 3-11 Autostop Q Apply log contents on secondary server JAMAICA. | 55 |
| 3-12 Start Q Apply on the primary server DENMARK | 57 |
| 3-13 Start Q Apply log contents on the primary server DENMARK | 58 |
| 3-14 Start Q Capture program on the primary server DENMARK | 58 |
| 3-15 Start Q Capture log contents on the primary server DENMARK | 58 |
| 3-16 Start Q Capture on the secondary server JAMAICA | 59 |
| 3-17 Start Q Apply on the secondary server JAMAICA | 59 |
| 3-18 Q Capture latency statistics | 60 |
| 3-19 Q End_to_End latency statistics | 61 |
| 3-20 Autostop Q Capture on the secondary server JAMAICA. | 65 |
| 3-21 Verify XMITQ on the secondary server JAMAICA is fully consumed. | 66 |
| 3-22 Verify RECVQ on the primary server DENMARK is fully consumed | 66 |
| 3-23 AsnQMFmt command output with “Last message in DB transaction” | 73 |
| 3-24 Verify RECVQ on the secondary server JAMAICA is fully consumed | 75 |
| 3-25 Stop Q Apply on the secondary server JAMAICA | 75 |
| 3-26 Autostop Q Capture on the primary server DENMARK | 78 |
| 3-27 Verify the Q Replication is up and running on primary and secondary servers 1/6 | 81 |
| 3-28 Verify the Q Replication is up and running on primary and secondary servers 2/6 | 81 |
| 3-29 Verify the Q Replication is up and running on primary and secondary servers 3/6 | 81 |
| 3-30 Verify the Q Replication is up and running on primary and secondary servers 4/6 | 81 |
| 3-31 Verify the Q Replication is up and running on primary and secondary servers 5/6 | 82 |

| | |
|--|-----|
| 3-32 Verify the Q Replication is up and running on primary and secondary servers 6/6 | 82 |
| 3-33 Autostop Q Apply on the primary server DENMARK | 87 |
| 3-34 Autostop Q Apply log contents on the primary server DENMARK | 88 |
| 3-35 Delete messages in XMITQ, RECVQ, ADMINQ, RESTARTQ on the primary server DENMARK | 88 |
| 3-36 Delete messages in XMITQ, RECVQ, ADMINQ, RESTARTQ on the secondary server JAMAICA | 92 |
| 3-37 Deactivate subscriptions on the primary server DENMARK | 97 |
| 3-38 Deactivate subscriptions on the secondary server JAMAICA | 97 |
| 3-39 Deactivate subscriptions on the primary server DENMARK | 99 |
| 3-40 Deactivate subscriptions on the secondary server JAMAICA | 99 |
| 3-41 Cold start Q Capture on the primary server DENMARK | 101 |
| 3-42 Cold start Q Capture log contents on the primary server DENMARK .. | 101 |
| 3-43 Cold start Q Capture on the secondary server JAMAICA | 101 |
| 3-44 Cold start Q Capture log contents on the secondary server JAMAICA .. | 101 |
| 3-45 Deactivate subscriptions on the primary server DENMARK | 107 |
| 3-46 Deactivate subscriptions on the secondary server JAMAICA | 108 |
| 3-47 Remove WebSphere MQ objects. | 114 |
| 3-48 The ASNCLP_drop_ctl.in to remove the Q replication control tables on JAMAICA | 115 |
| 3-49 Remove Q Capture and Q Apply logs | 115 |
| 4-1 setup_hadr_primary.sh script on the primary server DENMARK | 124 |
| 4-2 setup_hadr_standby.sh script on the secondary server JAMAICA | 125 |
| 4-3 Verify the source database cfg parameters on the standby server JAMAICA 1/2 | 125 |
| 4-4 Verify the source database cfg parameters on the standby server JAMAICA 2/2 | 125 |
| 4-5 start_hadr_standby.sh script on the secondary server JAMAICA | 128 |
| 4-6 deactivate_primary.sh script on the primary server DENMARK | 129 |
| 4-7 Verify db cfg setting on the primary server DENMARK | 130 |
| 4-8 Verify db cfg setting on the primary server DENMARK | 130 |
| 4-9 start_hadr_primary script on the primary server DENMARK | 133 |
| 4-10 Check HADR status on the primary server DENMARK | 133 |
| 4-11 Check HADR status on the standby server JAMAICA | 134 |
| 4-12 The update_alterate_server_primary.sh script on the primary server DENMARK | 135 |
| 4-13 The update_alterate_server_primary.sh script on the standby server JAMAICA | 135 |
| 4-14 LIST DB DIRECTORY on the primary server DENMARK | 136 |
| 4-15 LIST DB DIRECTORY on the standby server JAMAICA | 136 |
| 4-16 Catalog primary server DENMARK node on the client | 137 |
| 4-17 Catalog primary server DENMARK database SOURCE on the client .. | 137 |

| | |
|---|-----|
| 4-18 LIST DB DIRECTORY on the client | 137 |
| 4-19 Connect to SOURCE database from the client | 138 |
| 4-20 LIST DB DIRECTORY on the client | 138 |
| 4-21 get connection state | 139 |
| 4-22 db2 list applications on the primary server DENMARK | 139 |
| 4-23 Select statement issued from the client | 139 |
| 4-24 TAKEOVER HADR ON DATABASE SOURCE command on JAMAICA | 140 |
| 4-25 Select statement issued from the client | 140 |
| 4-26 db2diag.log contents on the client | 140 |
| 4-27 LIST DB DIRECTORY on the client | 142 |
| 4-28 Get Connection State on the client | 143 |
| 4-29 db2 list applications on the standby server JAMAICA | 143 |
| 4-30 Select statement issued from the client | 143 |
| 4-31 Create test tables on the primary server DENMARK | 146 |
| 4-32 Create test tables on the secondary server CLYDE | 146 |
| 4-33 Create test tables on the secondary server BONNIE | 147 |
| 4-34 Set up WebSphere MQ on MQ server SEVERN | 148 |
| 4-35 Start all the WebSphere MQ listener ports on SEVERN | 150 |
| 4-36 Configure WebSphere MQ client on CLYDE | 150 |
| 4-37 Configure WebSphere MQ client on BONNIE | 150 |
| 4-38 Configure WebSphere MQ client on DENMARK | 150 |
| 4-39 Configure WebSphere MQ client on JAMAICA | 150 |
| 4-40 Catalog databases on the ASNCLP machine | 151 |
| 4-41 Set up Java environment for userid "qrepladm" | 151 |
| 4-42 asncpl set up configuration | 152 |
| 4-43 Execute commands from a file | 152 |
| 4-44 Create the Q replication control tables on | 152 |
| 4-45 Create replication queue maps 1/2 | 153 |
| 4-46 Create replication queue maps 2/2 | 153 |
| 4-47 Create Q subscriptions for the test tables 1/2 | 154 |
| 4-48 Create Q subscriptions for the test tables 2/2 | 154 |
| 4-49 Catalog the primary server node and database on the secondary server CLYDE | 155 |
| 4-50 Catalog the primary server node and database on the secondary server BONNIE | 155 |
| 4-51 Create the password file on the secondary server CLYDE | 156 |
| 4-52 Create the password file on the secondary server BONNIE | 156 |
| 4-53 Start Q Apply on the secondary server CLYDE | 157 |
| 4-54 Start Q Apply log contents on the secondary server CLYDE | 158 |
| 4-55 Start Q Apply on the secondary server BONNIE | 158 |
| 4-56 Start Q Apply log contents on the secondary server BONNIE | 158 |
| 4-57 Start Q Capture on the primary server DENMARK | 159 |
| 4-58 Start Q Capture log contents on the primary server DENMARK | 159 |

| | | |
|-------|--|-----|
| 4-59 | Connect to the primary server DENMARK from the client machine. . . . | 160 |
| 4-60 | Connection state from the client machine | 160 |
| 4-61 | List database directory from the client machine. | 161 |
| 4-62 | Run a workload from the client machine | 161 |
| 4-63 | Stop Q Capture on the HADR primary server DENMARK. | 163 |
| 4-64 | Stop Q Capture stdout on the HADR primary server DENMARK | 163 |
| 4-65 | Stop Q Capture log contents on the HADR primary server DENMARK. | 163 |
| 4-66 | Check HADR status on the primary server DENMARK. | 164 |
| 4-67 | Check HADR status on the standby server JAMAICA. | 165 |
| 4-68 | Issue Takeover HADR on database source command on JAMAICA . . | 165 |
| 4-69 | Contents of the db2diag.log on the primary server DENMARK | 166 |
| 4-70 | Contents of the db2diag.log on the standby server JAMAICA. | 168 |
| 4-71 | Check HADR status on the primary server DENMARK. | 170 |
| 4-72 | Check HADR status on the standby server JAMAICA. | 171 |
| 4-73 | Start Q Capture on the new primary server JAMAICA. | 172 |
| 4-74 | Start Q Capture log contents on the new primary server JAMAICA . . | 172 |
| 4-75 | Start Q Capture on the new primary server JAMAICA. | 173 |
| 4-76 | Start Q Capture log contents on the new primary server JAMAICA . . | 173 |
| 4-77 | Run a workload on the client machine 1/2 | 174 |
| 4-78 | Run a workload on the client machine 2/2 | 175 |
| 4-79 | db2diag.log contents on the client machine | 176 |
| 4-80 | get connection state command from the client machine | 178 |
| 4-81 | LIST DB DIRECTORY command from the client machine | 178 |
| 4-82 | asncqcmd on the new primary server JAMAICA | 179 |
| 4-83 | Subscription states on the new primary server JAMAICA | 179 |
| 4-84 | asncqcmd on the secondary server CLYDE | 180 |
| 4-85 | Subscription states on the secondary server CLYDE. | 180 |
| 4-86 | asncqcmd on the secondary server BONNIE | 181 |
| 4-87 | Subscription states on the secondary server BONNIE | 181 |
| 4-88 | Contents of source tables on the primary server JAMAICA. | 182 |
| 4-89 | Contents of the target tables on the secondary server CLYDE | 183 |
| 4-90 | Contents of the target tables on the secondary server BONNIE | 184 |
| 4-91 | db2stop force command on the primary server DENMARK. | 188 |
| 4-92 | Check HADR status on the standby server JAMAICA. | 188 |
| 4-93 | Issue Takeover HADR on database source by force. | 189 |
| 4-94 | Contents of the db2diag.log on the standby server JAMAICA. | 189 |
| 4-95 | Check HADR status on the standby server JAMAICA. | 192 |
| 4-96 | Run a workload on the client 1/2 | 194 |
| 4-97 | Run a workload on the client 2/2 | 194 |
| 4-98 | Contents of source tables on the primary server JAMAICA. | 195 |
| 4-99 | Contents of the target tables on the secondary server CLYDE | 196 |
| 4-100 | Contents of the target tables on the secondary server BONNIE | 197 |
| 4-101 | Start DB2 on the old primary server DENMARK | 199 |

| | |
|--|-----|
| 4-102 Start HADR on database source as standby command on DENMARK | 200 |
| 4-103 Contents of the db2diag.log on the DENMARK | 200 |
| 4-104 Contents of the db2diag.log on JAMAICA | 204 |
| 4-105 Check HADR status on DENMARK | 205 |
| 4-106 Check HADR status on JAMAICA | 206 |
| 4-107 Stop Q Capture on JAMAICA | 206 |
| 4-108 Stop Q Capture stdout on the current primary server JAMAICA | 207 |
| 4-109 Stop Q Capture log contents on the current primary server JAMAICA | 207 |
| 4-110 Issue Takeover HADR on database source command on DENMARK | 208 |
| 4-111 Contents of the db2diag.log on DENMARK | 208 |
| 4-112 Contents of the db2diag.log on JAMAICA | 210 |
| 4-113 Check HADR status on DENMARK | 213 |
| 4-114 Check HADR status on JAMAICA | 213 |
| 4-115 Run a workload on the client 1/2 | 214 |
| 4-116 Run a workload on the client 2/2 | 215 |
| 4-117 db2diag.log contents on the client machine | 215 |
| 4-118 Connection state from the client machine | 217 |
| 4-119 List database directory from the client machine | 218 |
| 4-120 Contents of source tables on the primary server DENMARK | 219 |
| 4-121 Contents of target tables on the secondary server CLYDE | 220 |
| 4-122 Contents of target tables on the secondary server BONNIE | 221 |
| B-1 Conflict rule considerations | 232 |
| B-2 Conflict action considerations | 235 |
| B-3 Specifying OKSQLSTATES for a subscription using SQL | 237 |
| B-4 Error action considerations | 239 |
| B-5 Using Exceptions Table Formatter to view exceptions | 240 |
| B-6 Using SQL SELECT statements to view exceptions | 246 |
| B-7 Insert the independent/parent row and the same key exists - FORCE | 254 |
| B-8 Insert the independent /parent row and a row with the same key exists - IGNORE | 255 |
| B-9 Insert the child row and the row with the same key and its parent exists - FORCE | 257 |
| B-10 Insert the child row and the same key and the child's parent(s) exists - IGNORE | 258 |
| B-11 Insert child row but same key and child's parent missing | 259 |
| B-12 Insert the child row but the same key and child's parent is missing - FORCE 1/2 | 260 |
| B-13 Insert the child row but the same key and child's parent is missing - FORCE 2/2 | 261 |
| B-14 Insert child row but same key and child's parent is missing - IGNORE | 262 |
| B-15 Delete independent/parent/child row and same key missing - FORCE | 263 |
| B-16 Delete independent/parent/child row and same key missing - IGNORE | 264 |
| B-17 Delete independent row and same key exists | 265 |

| | |
|---|-----|
| B-18 Delete independent row and same key exists | 265 |
| B-19 Delete parent row and same key exists - value-based conflict - FORCE | 267 |
| B-20 Delete parent row and same key exists - SQL error exception | 267 |
| B-21 Delete the parent row and the same key exists - value-based conflict - IG- NORE | 268 |
| B-22 Update independent/parent/child row and same key missing - FORCE | 269 |
| B-23 Update independent/parent/child row and same key missing - IGNORE | 270 |
| B-24 Update independent/parent/child row and same key exists - value-based conflict - FORCE | 272 |
| B-25 Update independent/parent/child row and same key exists - SQL error ex- ception | 273 |
| B-26 Update independent/parent/child row and same key exists - value-based conflict - IGNORE | 274 |
| B-27 Update the independent/parent/child row and key exists - all match - SQL error | 275 |

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

z/OS®

AIX®


DB2 Universal Database™

DB2®

Informix®

IBM®

Redbooks™

Redbooks (logo) ™

WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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, and documents a step-by-step approach to implementing high availability scenarios involving bidirectional Q replication and HADR technologies.

It is important to note that this redbook uses the next release of Q replication in all of its scenarios. While we have not tested these scenarios with the current V8.3 release, we fully expect the scenarios and approaches documented here to apply.

It is aimed at an audience of IT architects and database administrators (DBA) responsible for developing high availability solutions on the AIX® platform.

This book is organized as follows:

- ▶ Chapter 1, "High availability concepts" on page 1 provides an introduction to high availability considerations in general, and describes some of the terminology used in this redbook.
- ▶ Chapter 2, "WebSphere Replication Server Q replication overview" on page 9 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 33 describes a step by step approach to performing failover and switchback in a bidirectional replication scenario involving two AIX servers.
- ▶ Chapter 4, "HADR and Q replication coexistence scenario" on page 117 describes a step by step approach to implementing an HADR 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 223 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 225 describes exception processing in Q replication.
- ▶ Appendix C, "Overview of HADR and DB2 Client Reroute" on page 301 provides a brief overview of HADR and DB2® Client Reroute.

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 master's 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 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's 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 worked in technical support for information management products for 4 years. She 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 Redbooks, such as *WebSphere Application Server V6: High Availability Solutions*, SG24-6688, for high availability concepts, and from IBM Silicon Valley Laboratory, Tony Lee's 2006 article "Comparison of logical High Availability solutions in DB2: HADR vs. Q Replication".

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

Sangam Racherla
International Technical Support Organization, San Jose 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



High availability concepts

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

1.1 Introduction

This redbook describes high availability (HA) implementations on the AIX platform using WebSphere Information Integrator Q Replication and HADR technologies. It is therefore desirable to define the term “availability” and the various levels of availability that exist.

System¹ 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:

$$A = \text{MTBF} / (\text{MTBF} + \text{MTTR})$$

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, and
- ▶ 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:

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

- ▶ 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 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 on page 4. 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.

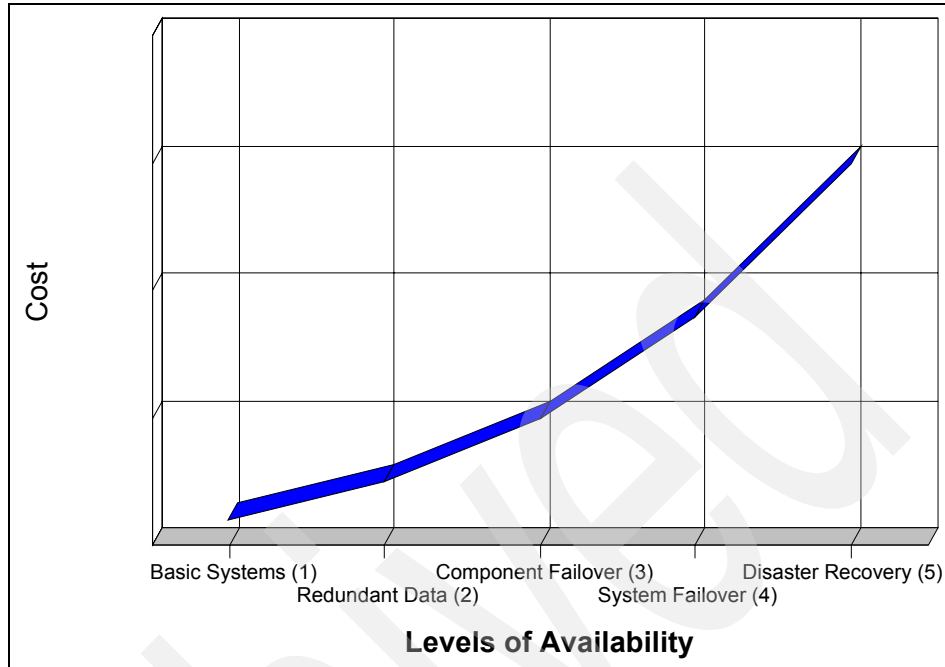


Figure 1-1 Levels of availability and costs

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. Neither 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%.

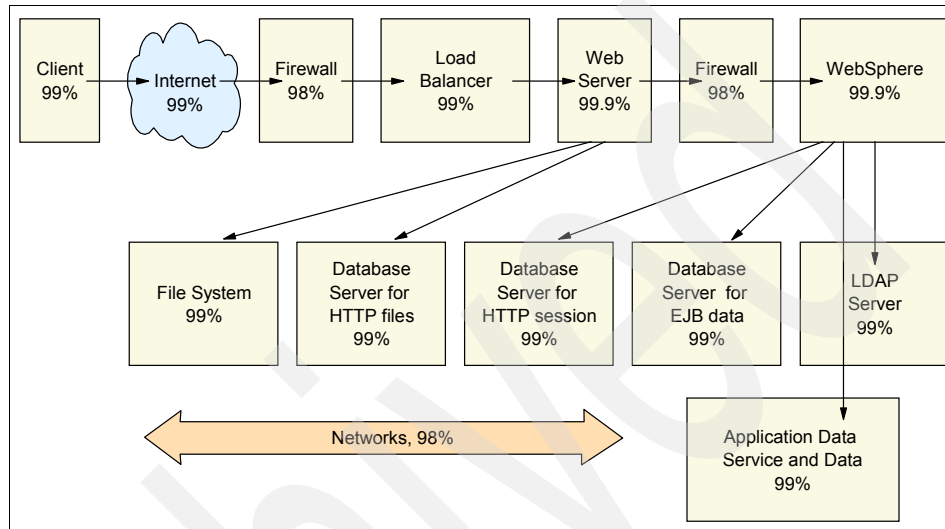


Figure 1-2 Availability chains

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.

In system failover typically, 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/Active² mutual takeover or Active/Passive³ takeover.

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

² Both the primary and 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.

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.
- ▶ 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 |

| 9s | Percentage of uptime | Downtime per year | Downtime per week | Downtime per day |
|----------|----------------------|--------------------|-------------------|------------------|
| 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)

It 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.



WebSphere Replication

Server Q replication overview

In this chapter we provide an overview of Q replication and its architecture and processing flow. We 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 was previously called WebSphere Information Integrator Replication Edition.

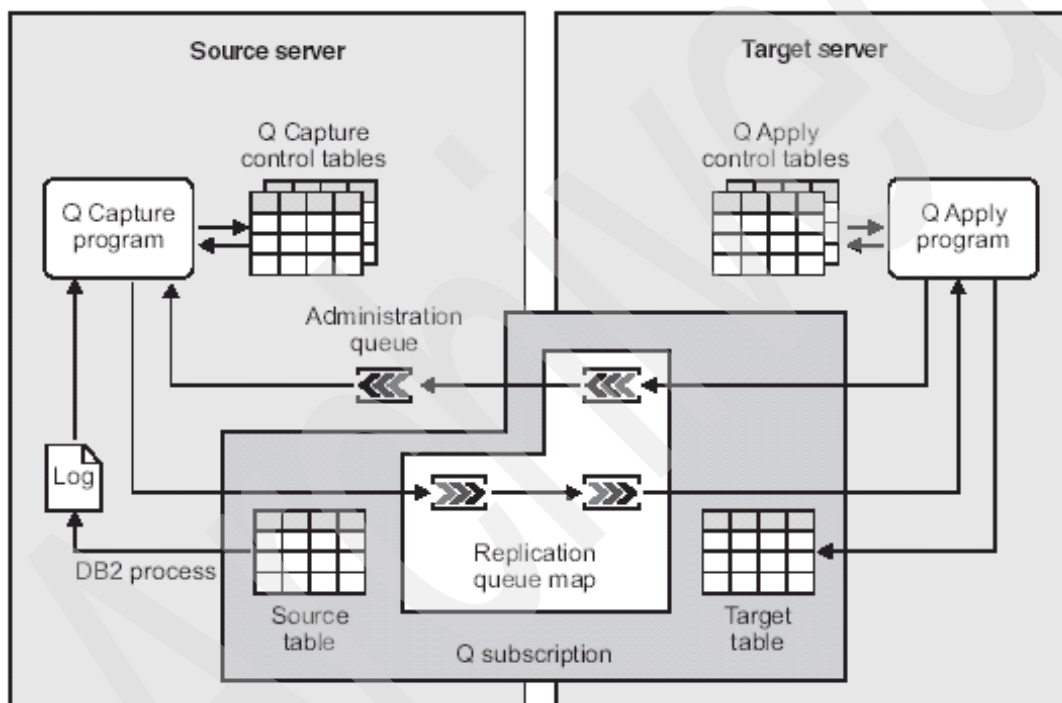


Figure 2-1 A simple Q replication configuration

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 (CCD¹) 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.

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.

¹ 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, incomplete CCD table, and non condensed CCD table.

- ▶ 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 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.

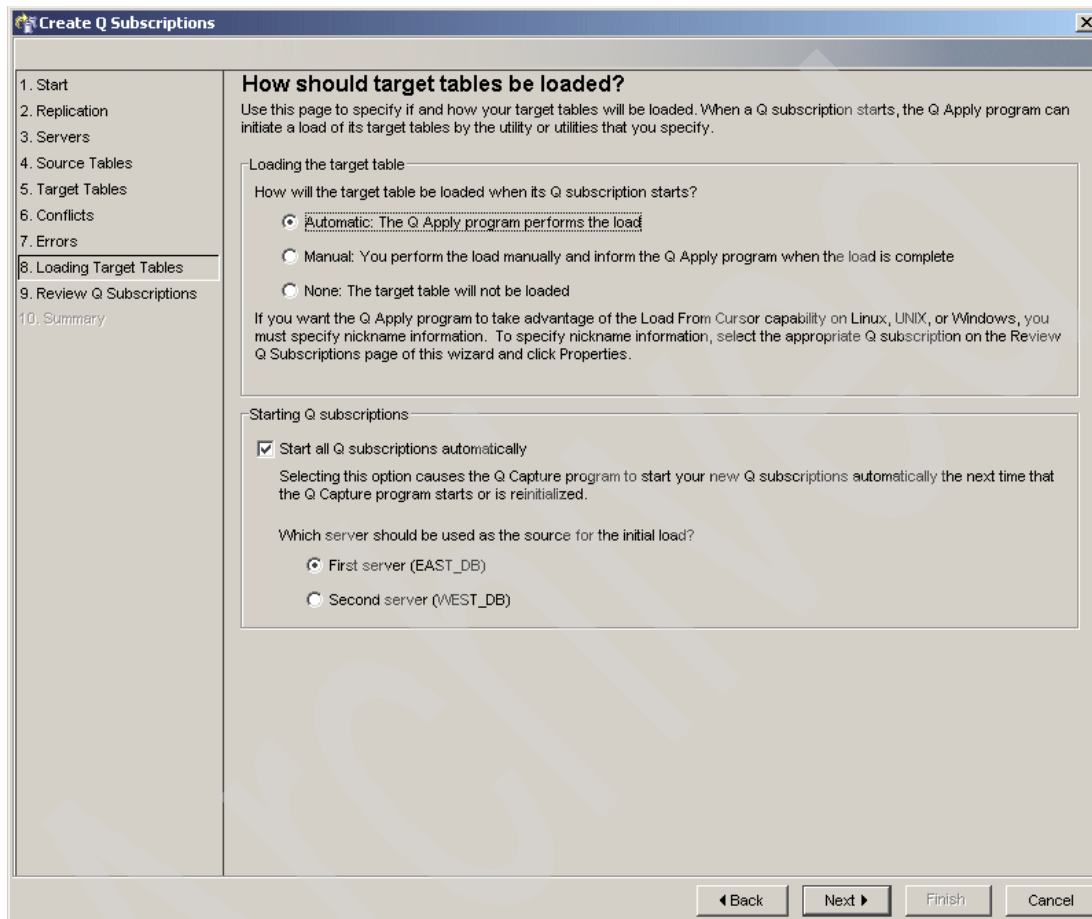


Figure 2-2 Specifying how target table is to be loaded for given Q subscription

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 though 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
```

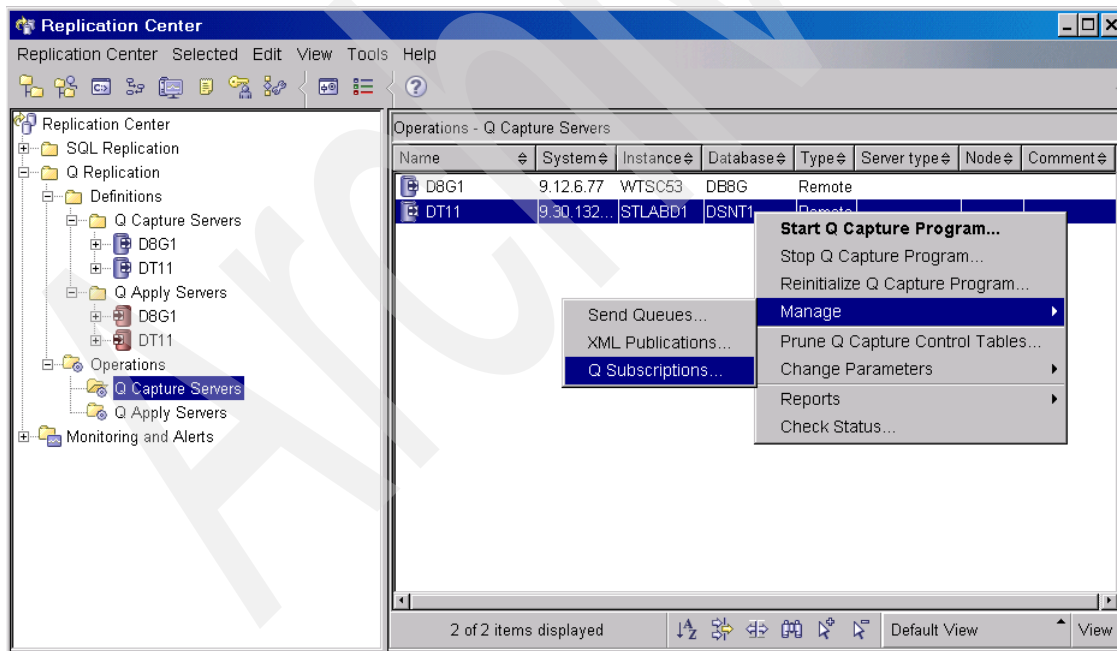


Figure 2-3 Activating Q subscription via the Replication Center

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.

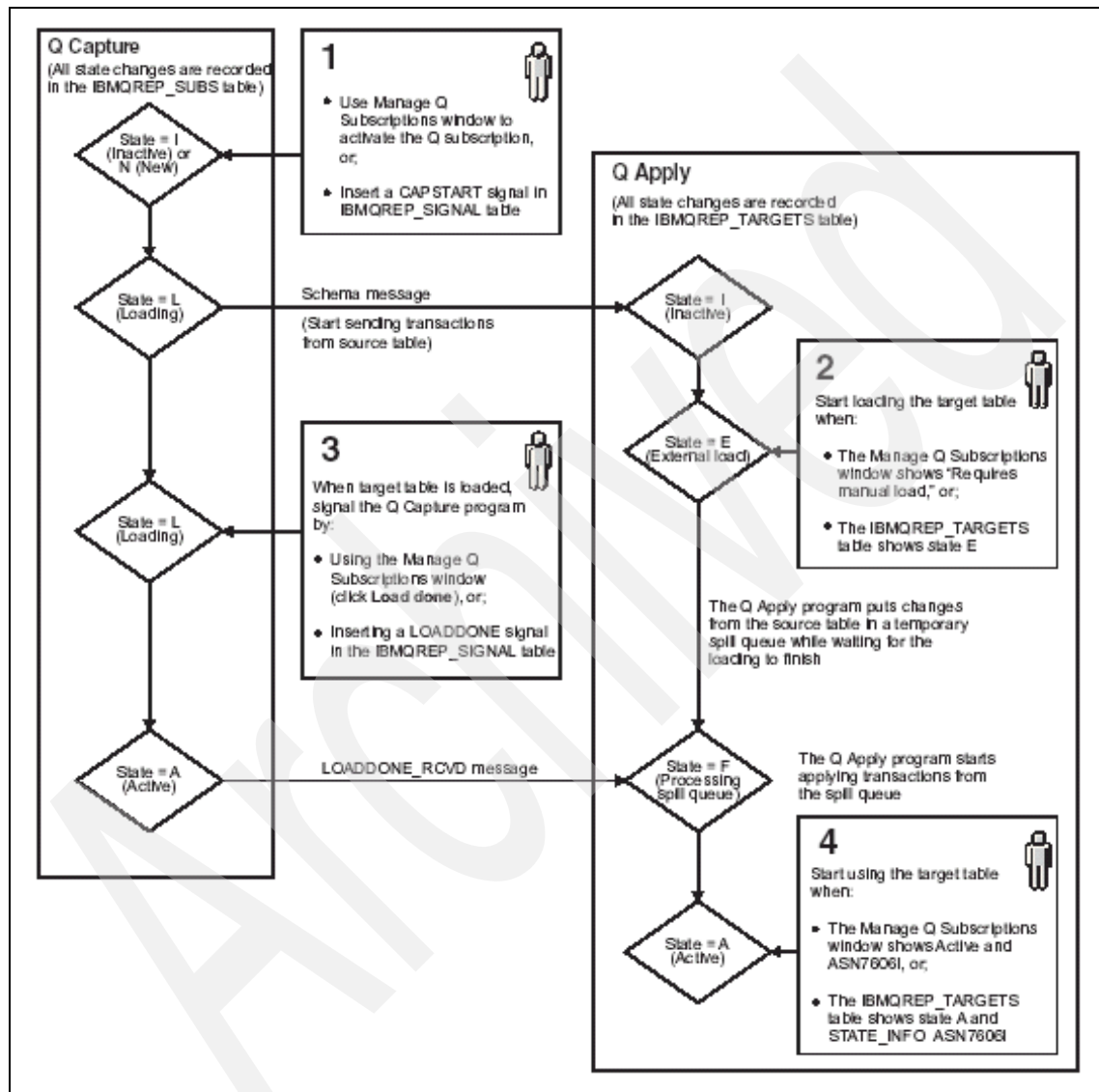


Figure 2-4 Manual load processing flow in unidirectional Q replication

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. In Figure 2-4 on page 16, 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 2-5 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.

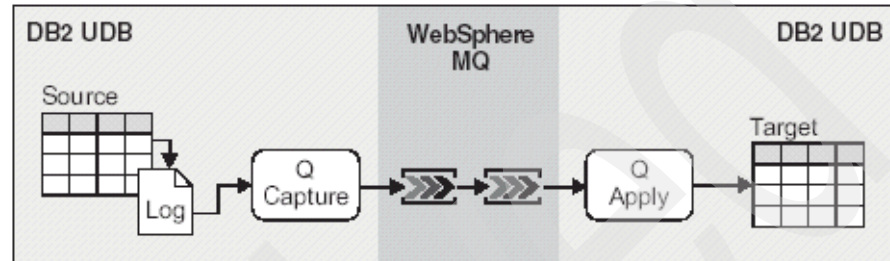


Figure 2-5 Ongoing replication flow

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 — 500 ms 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 DB2 Information Center

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

WebSphere MQ

WebSphere MQ ensures that every message² 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.

² Q replication messages are persistent.

- ▶ *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.
- ▶ *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 which 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 conflicts³.

Note: As of fixpak 10, you can now replicate to target tables without a primary key or unique constraint, 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 (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 order⁴ that they were committed at the source. By default, the Q Apply program applies transactions in parallel using multiple agent threads. Such parallelism is important when many changes are replicated from the source server.

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.

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 lets 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 <ul style="list-style-type: none"> ▶ "Master" wins conflict ▶ Not detecting all conflicts is acceptable (such as LOB conflicts) ▶ Latest update wins conflict ▶ Conflict detection 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 <ul style="list-style-type: none"> ▶ 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 on page 30 shows how these statistics may be computed using SQL.

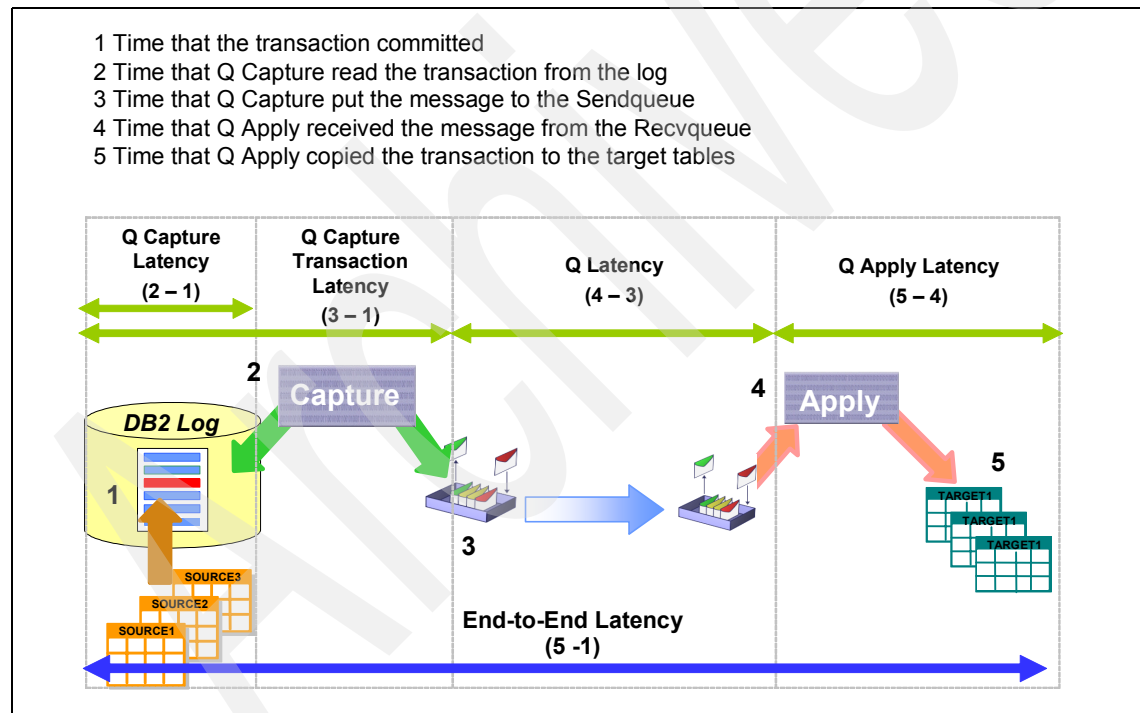


Figure 2-6 Latency statistics collected

| 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 are 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 value 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.

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.
- ▶ 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*, SC34-6068, 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.

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 AIX servers.

The topics covered are:

- ▶ Business requirements
- ▶ 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, Muntok, 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, Muntok is required to make the hot site available for read only access for reporting purposes to take 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 resynchronized 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 2.3, “Choosing a particular Q replication topology” on page 28, and the business requirements defined in 3.2, “Business requirement” on page 34, the bidirectional replication topology is appropriate for Muntok. for the following reasons

1. Less stringent requirements for failover time (within 30 minutes instead of an instantaneous requirement which would tilt in favor of peer-to-peer)
2. Non-dedicated backup server requirement
3. Absence of conflicts¹ 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.

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.

5. Muntok's 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 Muntok bidirectional replication topology.

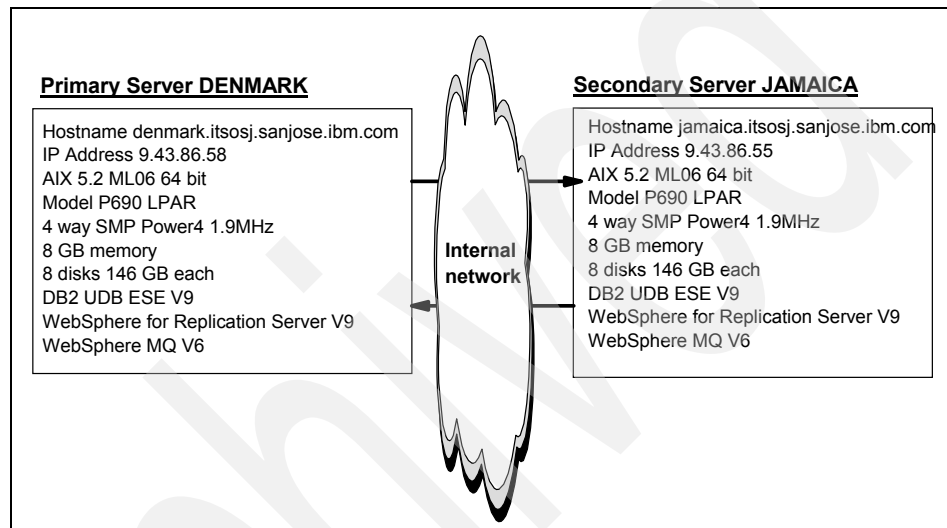


Figure 3-1 Bidirectional replication topology configuration

We installed a set of four tables with referential integrity constraints defined between some of them as shown in Figure 3-2 on page 37. The combination of delete RESTRICT and delete CASCADE options were essential to describing the conflict and SQL error exceptions. The OPTIONS table has a CLOB column.

Note: The identity column in the TRAN 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 on page 37.

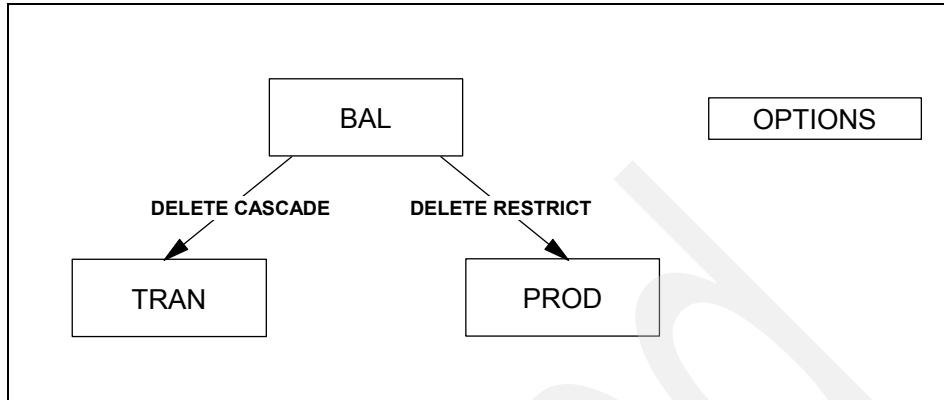


Figure 3-2 Tables used in the bidirectional replication scenario

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

 -- DDL Statements for table "DB2INST9"."PRODUCT"

```

CREATE TABLE "DB2INST9"."PROD" (
  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
)
IN "USERSPACE1" ;
  
```

```

ALTER TABLE "DB2INST9"."PROD" DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
  
```

-- DDL Statements for primary key on Table "DB2INST9"."PROD"

```

CREATE TYPE 2 UNIQUE INDEX IDX_PK_PROD ON PROD(ACCT_ID, PROD_TYPE);
ALTER TABLE "DB2INST9"."PROD"
  ADD CONSTRAINT "PK_PORD" PRIMARY KEY(ACCT_ID, PROD_TYPE);
  
```

 -- DDL Statements for table "DB2INST9"."TRAN"

```

CREATE TABLE "DB2INST9"."TRAN" (
  ACCT_ID INTEGER NOT NULL
, TRAN_ID INTEGER GENERATED BY DEFAULT AS IDENTITY
  (START WITH $IDSTART 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
)
IN "USERSPACE1" ;

ALTER TABLE "DB2INST9"."TRAN" DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;

-- DDL Statements for primary key on Table "DB2INST9"."TRAN"

CREATE TYPE 2 UNIQUE INDEX IDX_PK_TRAN ON TRAN(ACCT_ID, TRAN_ID);
ALTER TABLE "DB2INST9"."TRAN"
  ADD CONSTRAINT "PK_TRAN" PRIMARY KEY
    (ACCT_ID, TRAN_ID);

-----
-- DDL Statements for table "DB2INST9"."BAL"
-----

CREATE TABLE "DB2INST9"."BAL" (
  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
)
IN "USERSPACE1" ;

ALTER TABLE "DB2INST9"."BAL" DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;

-- DDL Statements for primary key on Table "DB2INST9"."BAL"

CREATE TYPE 2 UNIQUE INDEX IDX_PK_BAL ON BAL(ACCT_ID);
ALTER TABLE "DB2INST9"."BAL"
  ADD CONSTRAINT "PK_BAL" PRIMARY KEY (ACCT_ID);

-----
-- DDL Statements for table "DB2INST9"."OPTIONS"
-----

CREATE TABLE "DB2INST9"."OPTIONS" (
  "OPTION_ID" INTEGER NOT NULL,
  "NOTES" CLOB(4765) NOT LOGGED NOT COMPACT NOT NULL ,
  "UPDATED" DATE NOT NULL )
IN "USERSPACE1" ;

ALTER TABLE "DB2INST9"."OPTIONS" DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;

-- DDL Statements for primary key on Table "DB2INST9"."OPTIONS"

```

```

CREATE TYPE 2 UNIQUE INDEX IDX_OPTIONS
ON OPTIONS(OPTION_ID);
ALTER TABLE "DB2INST9"."OPTIONS"
  ADD CONSTRAINT "PK_OPTIONS" PRIMARY KEY
    ("OPTION_ID");

-- DDL Statements for foreign keys on Table "DB2INST9"."TRAN"
ALTER TABLE "DB2INST9"."TRAN"
  ADD CONSTRAINT "FK_TRAN" FOREIGN KEY ("ACCT_ID")
    REFERENCES "DB2INST9"."BAL" ("ACCT_ID")
    ON DELETE CASCADE
    ENFORCED
    ENABLE QUERY OPTIMIZATION;

-- DDL Statements for foreign keys on Table "DB2INST9"."PROD"
ALTER TABLE "DB2INST9"."PROD"
  ADD CONSTRAINT "FK_PROD" FOREIGN KEY ("ACCT_ID")
    REFERENCES "DB2INST9"."BAL" ("ACCT_ID")
    ON DELETE RESTRICT
    ENFORCED
    ENABLE QUERY OPTIMIZATION;

```

Figure 3-3 on page 40 provides a high-level overview of the various objects involved in implementing a bidirectional replication topology for Muntok. In Figure 3-3 on page 40, 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 on page 40 has the appearance of two sets so that the flow of data and messages between the two servers is easily understood.

Attention: The set up of the bidirectional Q replication environment described in Figure 3-1, Figure 3-2 and Figure 3-3 were performed using scripts which can be downloaded from the Redbooks Web site:

<ftp://www.redbooks.ibm.com/redbooks/SG247216/>

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 redbook 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.

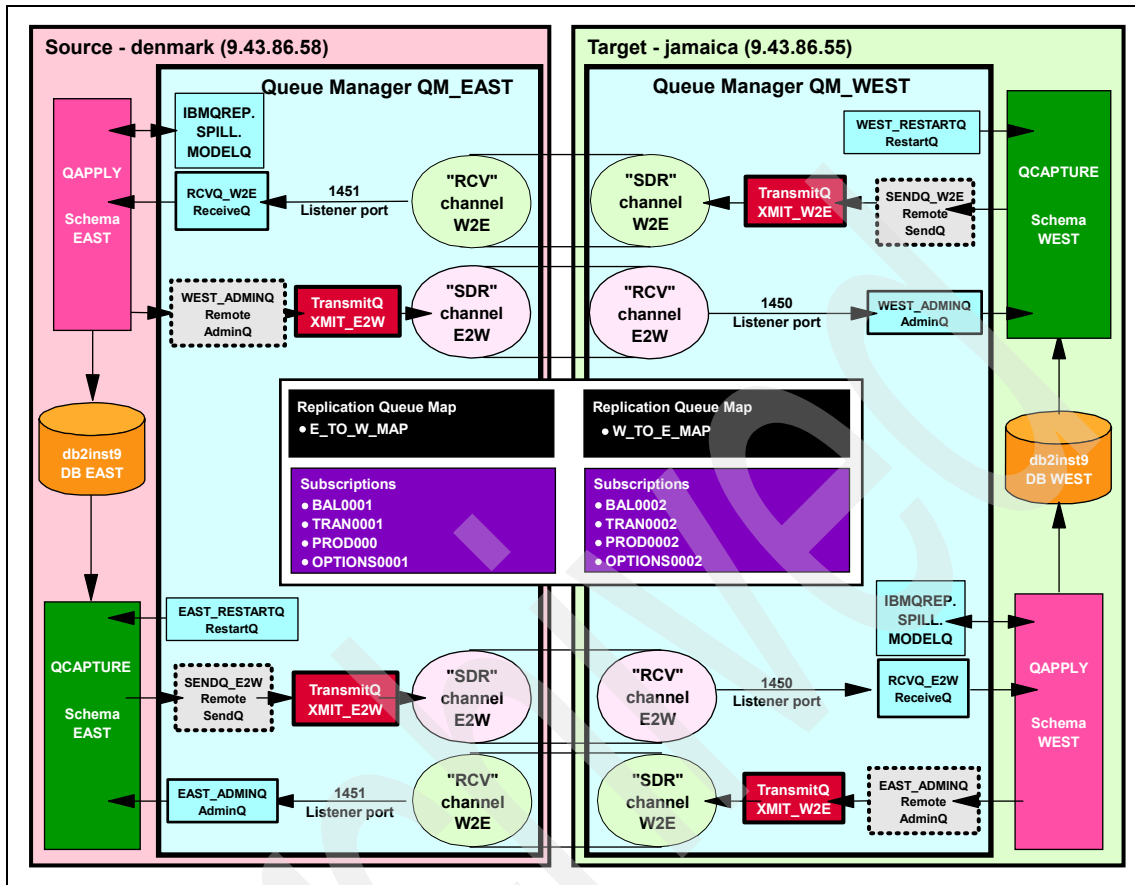


Figure 3-3 Bidirectional replication topology objects overview

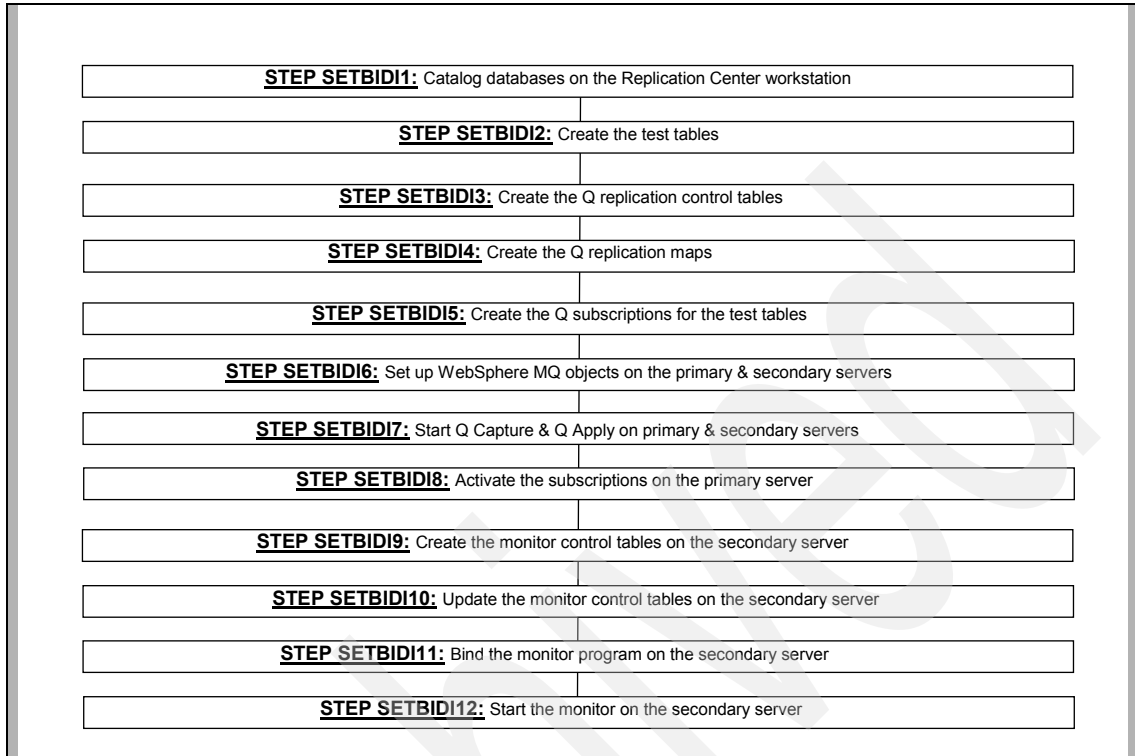


Figure 3-4 Overview of set up steps for the bidirectional replication environment

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.

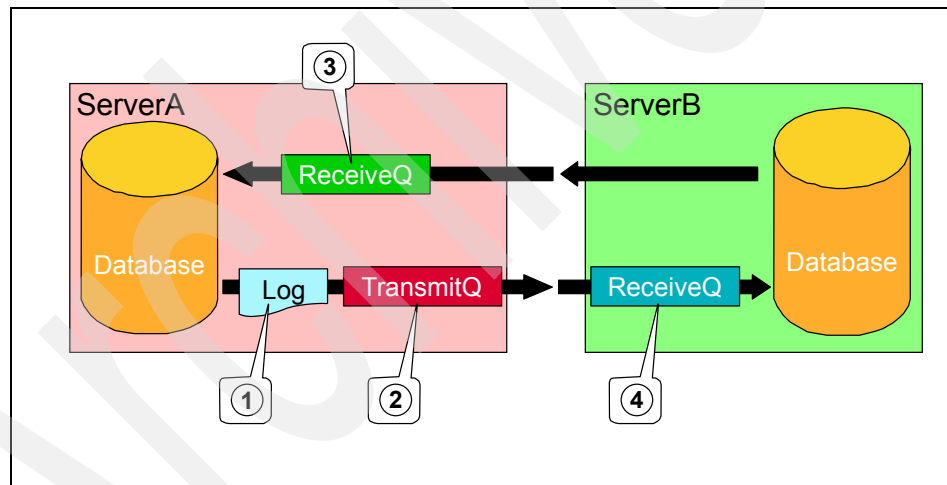


Figure 3-5 Data loss and potential sources of conflicts

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 on page 42). 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 on page 42) 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 Muntok 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 42), or messages in the WebSphere MQ queue that have not been transmitted to the secondary server (item 2 in Figure 3-5 on page 42), or messages in the receive queue on the secondary server that have not been drained (item 4 in Figure 3-5 on page 42) 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 Muntok’s bidirectional topology, this involves restoring DENMARK as the primary server where application updates occur, and JAMAICA 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 financial applications.

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 42). 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 42). 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 on page 42) 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 on page 46 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 |
| 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 — 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 subscription(s) is 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 — 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 subscription(s) is 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 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 on page 46 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 D² 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

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

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 on page 49 summarizes the failover and switchback options discussed earlier.

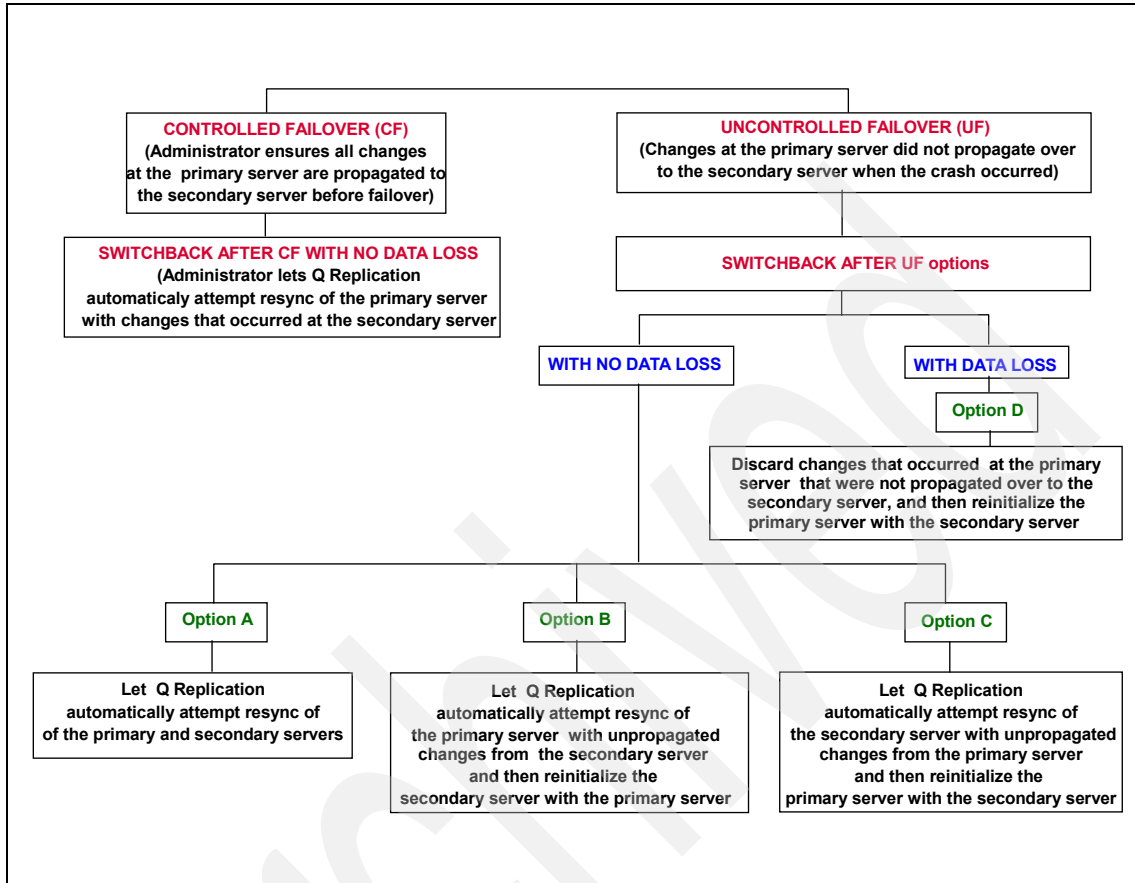


Figure 3-6 Bidirectional replication high availability failover and switchback options

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 on page 51 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.
- ▶ Archive log files on the secondary server are kept on DB2 ARCHIVE LOGPATH, and are not removed during the outage of the primary server.
- ▶ 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.

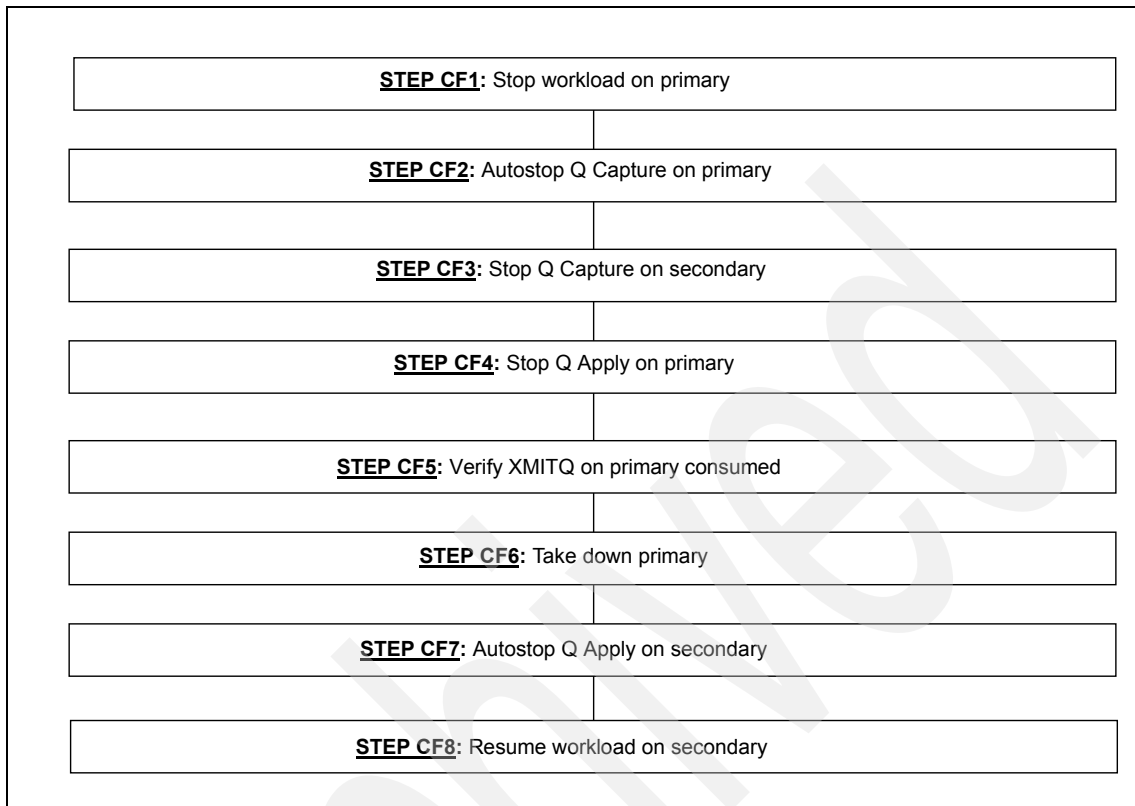


Figure 3-7 Overview of controlled failover (CF) steps

The steps shown in Figure 3-7 are 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 redbook.

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 on page 52 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. You may also monitor the Q Capture process by issuing the **ps -ef | grep asnqcap | grep -v grep** command.

Example 3-2 Autostop Q Capture on the primary server DENMARK

asnqccmd capture_server=EAST capture_schema=EAST chgparms autostop=Y

2006-02-23-19.26.38.237775 ASN0523I "AsnQCmd" : "EAST" : "Initial" : The CHGPparms command response: "AUTOSTOP" has been set to "Y".

----Contents of stdout

2006-02-23-19.26.45.719578 ASN0573I "Q Capture" : "EAST" : "Initial" : The program was stopped.

[asnqwk] Leaving the asnqwk thread:

[asnqwk] total database transactions captured = 0

[asnqwk] total MQSeries transactions = 0

[asnqwk] total DB transactions published = 0

[asnqwk] total time WAITING for logrd = 45021 seconds

[asntxrd] Leaving logtxrd reader thread:

[asntxrd] 0 message(s) in notification queue

[asntxrd] transmgr memory in use = 0 bytes

[asntxrd] totalLogReadTime = 0.021 second(s)

[asntxrd] total logrd SLEEP time = 50.000 second(s)

[asntxrd] totalLogRecsRead = 0

[asntxrd] Queued a thread Termination Notification.

[waitForLogrdEnd] The logrd thread terminated with rc: 0

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

.....
2006-02-23-19.26.37.702779 <asnqwk> ASN7020I "Q Capture" : "EAST" : "WorkerThread" : The program reached the end of the active log and terminated because the AUTOSTOP option is specified.

2006-02-23-19.26.37.702879 <asnqwk> ASN7109I "Q Capture" : "EAST" : "WorkerThread" : At program termination, the highest log sequence number of a successfully processed transaction is "43FE:5349:0000:0001:0000" and the lowest log sequence number of a transaction still to be committed is "0000:0000:0000:599F:42E3".

.....
2006-02-23-19.26.45.719578 <asnqcap::main> ASN0573I "Q Capture" : "EAST" : "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. You may also monitor the Q Capture process by issuing the **ps -ef | grep asnqcap | grep -v grep** command.

Example 3-4 Stop Q Capture on the secondary server JAMAICA

asnqcmd capture_server=WEST capture_schema=WEST STOP

2006-02-23-17.53.51.004223 ASN0522I "AsnQCcmd" : "WEST" : "Initial" : The program received the "STOP" command.

----Contents of stdout

```
[asnqwk] Leaving the asnqwk thread:
[asnqwk] total database transactions captured = 0
[asnqwk] total MQSeries transactions          = 0
[asnqwk] total DB transactions published      = 0
[asnqwk] total time WAITING for logrd        = 66492 seconds
[asntxrd] Leaving logtxrd reader thread:
[asntxrd] 0 message(s) in notification queue
[asntxrd] transmgr memory in use              = 0 bytes
[asntxrd] totalLogReadTime                    = 0.023 second(s)
[asntxrd] total logrd SLEEP time              = 70.000 second(s)
[asntxrd] totalLogRecsRead                    = 0
[asntxrd] Queued a thread Termination Notification.
[waitForLogrdEnd] The logrd thread terminated with rc: 0
```

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

.....
 2006-02-23-17.53.50.888511 <asnqwk> ASN7109I "Q Capture" : "WEST" :
 "WorkerThread" : At program termination, the highest log sequence number of a
 successfully processed transaction is "43FE:37A5:0000:0004:0000" and the lowest
 log sequence number of a transaction still to be committed is
 "0000:0000:0000:1433:E09E".

2006-02-23-17.53.57.406969 <asnqcap::main> ASN0573I "Q Capture" : "WEST" :
 "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 on page 54 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

Example 3-6 Stop Q Apply on the primary server DENMARK

2006-02-23-20.00.14.365576 ASN0522I "AsnQAcmd" : "EAST" : "Initial" : The program received the "STOP" command.

```
.....  
2006-02-23-20.13.11.565933 <browser::~~browser> ASN8999D "Q Apply" : "EAST" :  
"BR00000" : browser for queue 'RCVQ_W2E' found max msgID  
'5152455043FE374200000000000000000000000000000000000000000000000000000000000'  
'5152455043FE374200000000000000000000000000000000000000000000000000000000000'  
IBMQREP_DONEMSG table on shutdown.  
.....  
2006-02-23-20.13.13.531252 <asnqapp::main> ASNO573I "Q Apply" : "EAST" :  
"Initial" : The program was stopped.
```

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. Example 3-8 shows the WebSphere MQ command to determine that the transmit queue is empty. The value of the CURDEPTH field is zero indicating that the transmit queue XMIT_E2W on the primary server has been fully consumed.

```

1 : dis ql(XMIT_E2W) curdepth
AMQ8409: Display Queue details.
      QUEUE(XMIT_E2W)                                TYPE(QLLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.

```

STEP CF6: Take down primary

The primary server can now be taken offline by issuing the **db2stop** command as shown in Example 3-9.

Example 3-9 Take down the primary server DENMARK

db2stop

```
02/23/2006 20:20:31    0    0    SQL1064N  DB2STOP processing was successful.  
SQL1064N  DB2STOP processing was successful.
```

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 3-10 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 3-11. You may also monitor the Q Apply process by issuing the **ps -ef | grep asnqapp | grep -v grep** command.

Example 3-10 Autostop Q Apply on the secondary server JAMAICA

```
asnqacmd apply_server=WEST apply_schema=WEST chgparms autostop=Y
```

```
2006-02-23-18.27.07.622321 ASN0522I  "AsnQAcmd" :  "WEST" : "Initial" : The program received  
the "STOP" command.
```

Example 3-11 Autostop Q Apply log contents on secondary server JAMAICA

```
.....  
2006-02-23-18.27.40.298206 <brwzMain> ASN8999D  "Q Apply" : "WEST" : "BR00000"  
: Browser for queue 'RCVQ_E2W' will be suspended because AUTOSTOP option was  
specified.  
2006-02-23-18.27.42.298882 <brwzMain> ASN7590I  "Q Apply" : "WEST" : "BR00000"  
: The Q Apply program stopped reading from the queue "RCVQ_E2W" for replication  
queue map "E_TO_W_MAP". Reason code: "0".  
.....  
2006-02-23-18.27.45.300088 <browser::~~browser> ASN8999D  "Q Apply" : "WEST" :  
"BR00000" : browser for queue 'RCVQ_E2W' found max msgID  
'5152455043FE527E00000000000000000000000000000046' after cleaning  
IBMQREP_DONEMSG table on shutdown.  
.....  
2006-02-23-18.27.49.180806 <asnqapp::main> ASN0573I  "Q Apply" : "WEST" :  
"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 redbook.

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 3-8 on page 57 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.
- ▶ Archive log files on the secondary server are kept on DB2 ARCHIVE LOGPATH, and are not removed during the outage of the primary server.
- ▶ 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.

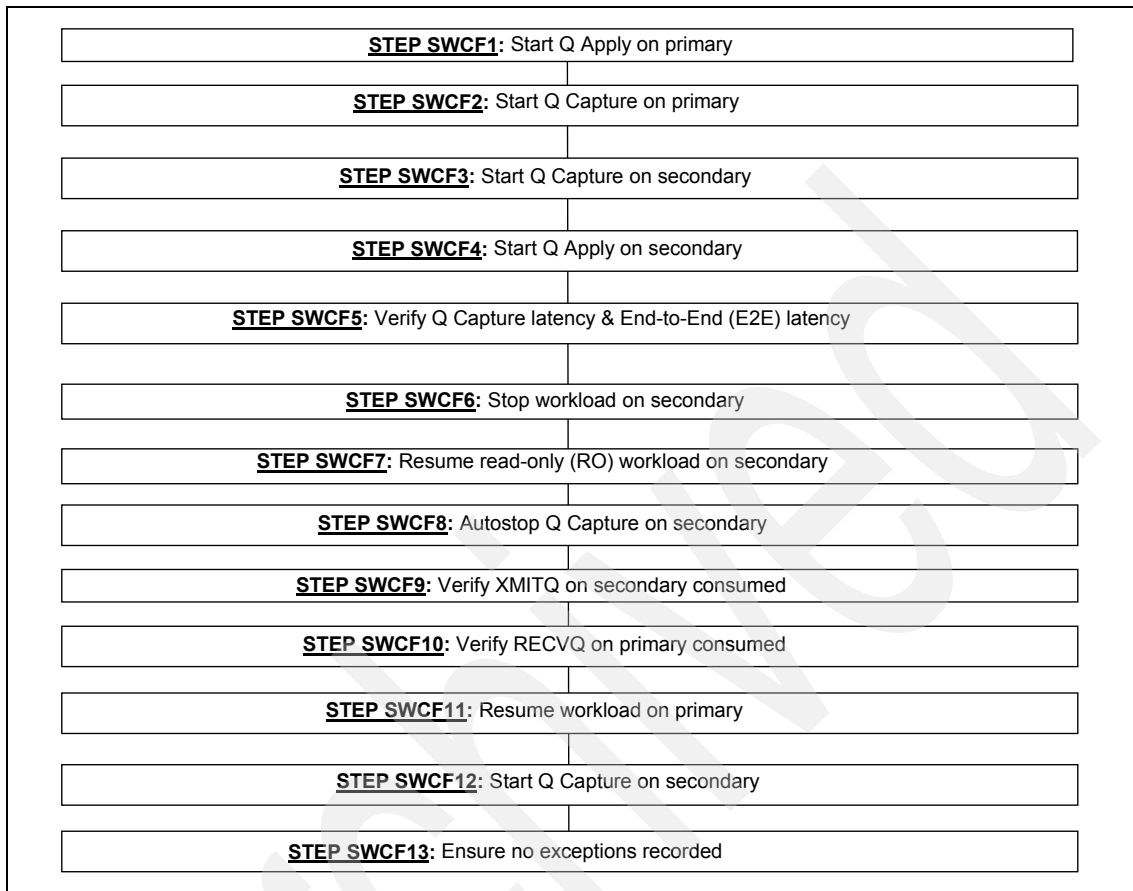


Figure 3-8 Overview of switchback after controlled failover (CF) steps

The steps shown in Figure 3-8 are described briefly in the following sections.

STEP SWCF1: Start Q Apply on primary

Start Q Apply on the primary server as shown in Example 3-12, the output of which indicates that the Q Apply program has started processing the receive queue RCVQ_W2E. 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 3-13 on page 58. You may also monitor the Q Apply process by issuing the `ps -ef | grep asnqapp | grep -v grep` command.

Example 3-12 Start Q Apply on the primary server DENMARK

```
asnqapp apply_server=EAST apply_schema=EAST apply_path=/DB2
```

2006-02-23-20.51.52.605805 ASN7526I "Q Apply" : "EAST" : "BR00000" : The Q Apply program has started processing the receive queue "RCVQ_W2E" for replication queue map "W_TO_E_MAP".

Example 3-13 Start Q Apply log contents on the primary server DENMARK

.....
2006-02-23-20.51.52.605805 <brwzMain> ASN7526I "Q Apply" : "EAST" : "BR00000"
: The Q Apply program has started processing the receive queue "RCVQ_W2E" for
replication queue map "W_TO_E_MAP".
.....
2006-02-23-20.51.52.630851 <appAgntMain> ASN8999D "Q Apply" : "EAST" :
"BR00000AG008" : agent 008 started for queue "RCVQ_W2E"

STEP SWCF2: Start Q Capture on primary

Start Q Capture on the primary server as shown in Example 3-14, the output of which indicates that the Q Capture program initialized successfully. You can monitor the contents of the Q Capture log which shows the Q Capture program initializing successfully as shown in Example 3-15. You may also monitor the Q Capture process by issuing the `ps -ef | grep asnqcap | grep -v grep` command.

Example 3-14 Start Q Capture program on the primary server DENMARK

asnqcap capture_server=EAST capture_schema=EAST capture_path=/DB2

2006-02-23-20.54.32.476768 ASN7000I "Q Capture" : "EAST" : "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.
2006-02-23-20.54.32.492714 ASN0572I "Q Capture" : "EAST" : "WorkerThread" : The program
initialized successfully.

Example 3-15 Start Q Capture log contents on the primary server DENMARK

.....
2006-02-23-20.54.31.976860 <asnParmClass::printParms> ASN0529I "Q Capture" :
"EAST" : "Initial" : The value of "STARTMODE" was set to "WARMSI" at startup by
the following method: "PARAMETERS TABLE".
.....
2006-02-23-20.54.32.476768 <subMgr::processSubscriptions> ASN7000I "Q Capture"
: "EAST" : "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.
2006-02-23-20.54.32.492653 <waitForLogrdInit> ASN7108I "Q Capture" : "EAST" :
"WorkerThread" : At program initialization, the highest log sequence number of

a successfully processed transaction is "43FE:5349:0000:0001:0000" and the lowest log sequence number of a transaction still to be committed is "0000:0000:0000:599A:7C69".

2006-02-23-20.54.32.492714 <asnqwk> ASN0572I "Q Capture" : "EAST" : "WorkerThread" : The program initialized successfully.

STEP SWCF3: Start Q Capture on secondary

Start Q Capture on the secondary server as shown in Example 3-16, the output of which indicates that the Q Capture program initialized successfully. 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 3-15 on page 58. You may also monitor the Q Capture process by issuing the `ps -ef | grep asnqcap | grep -v grep` command.

Example 3-16 Start Q Capture on the secondary server JAMAICA

asnqcap capture_server=WEST capture_schema=WEST capture_path=/DB2

2006-02-23-19.03.53.760069 ASN7000I "Q Capture" : "WEST" : "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.

2006-02-23-19.03.53.767426 ASN0572I "Q Capture" : "WEST" : "WorkerThread" : The program initialized successfully.

STEP SWCF4: Start Q Apply on secondary

Start Q Apply on the secondary server as shown in Example 3-17, the output of which indicates that the Q Apply program has started processing the receive queue RCVQ_E2W. 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 3-13 on page 58. You may also monitor the Q Apply process by issuing the `ps -ef | grep asnqapp | grep -v grep` command.

Example 3-17 Start Q Apply on the secondary server JAMAICA

asnqapp apply_server=WEST apply_schema=WEST apply_path=/DB2

2006-02-23-19.06.04.756272 ASN7526I "Q Apply" : "WEST" : "BR00000" : The Q Apply program has started processing the receive queue "RCVQ_E2W" for replication queue map "E_TO_W_MAP".

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 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:

- ▶ Using SQL as shown in details command as shown in Example 3-18 and Example 3-19 on page 61.
- ▶ Replication Center GUI as shown in Figure 3-9 on page 62 through Figure 3-12 on page 64.
- ▶ **asnlqacmd** show details command (not shown here).
- ▶ Live Monitor tool invoked via the **QApplyMonitor** command on a Windows workstation (not shown here).

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 3-18 Q Capture latency statistics

CONNECT TO WEST

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = QREPLADM
Local database alias = WEST

SELECT MONITOR_TIME, (MONITOR_TIME - CURRENT_LOG_TIME) AS CAP_LATENCY FROM WEST.IBMQREP_CAPMON ORDER BY MONITOR_TIME DESC FETCH FIRST 10 ROWS ONLY

| MONITOR_TIME | CAP_LATENCY |
|----------------------------|-------------|
| 2006-02-28-11.38.00.845677 | 2.340186 |
| 2006-02-28-11.37.30.802769 | 2.327392 |
| 2006-02-28-11.37.00.758829 | 2.314353 |
| 2006-02-28-11.36.30.709443 | 2.278093 |
| 2006-02-28-11.36.00.660067 | 2.242049 |
| 2006-02-28-11.35.30.615506 | 2.210683 |

6 record(s) selected.

Example 3-19 Q End_to_End latency statistics

CONNECT TO EAST

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = QREPLADM
Local database alias = EAST

**SELECT MONITOR_TIME, END2END_LATENCY, (END2END_LATENCY - QLATENCY -
APPLY_LATENCY) AS CAP_TX_LATENCY , TRANS_APPLIED AS TRANS , ROWS_APPLIED AS
ROWS FROM EAST.IBMQREP_APPLYMON WHERE ROWS_APPLIED<>0 ORDER BY MONITOR_TIME
DESC FETCH FIRST 5 ROWS ONLY**

| MONITOR_TIME | END2END_LATENCY | CAP_TX_LATENCY | TRANS | ROWS |
|----------------------------|-----------------|----------------|-------|------|
| 2006-02-28-11.37.46.121997 | 3582 | 3112 | 29 | 493 |
| 2006-02-28-11.37.16.121637 | 3755 | 3284 | 28 | 476 |
| 2006-02-28-11.36.46.121275 | 3618 | 3132 | 28 | 476 |
| 2006-02-28-11.36.16.121175 | 3801 | 3331 | 28 | 476 |
| 2006-02-28-11.35.46.120807 | 3601 | 3132 | 28 | 476 |

5 record(s) selected.

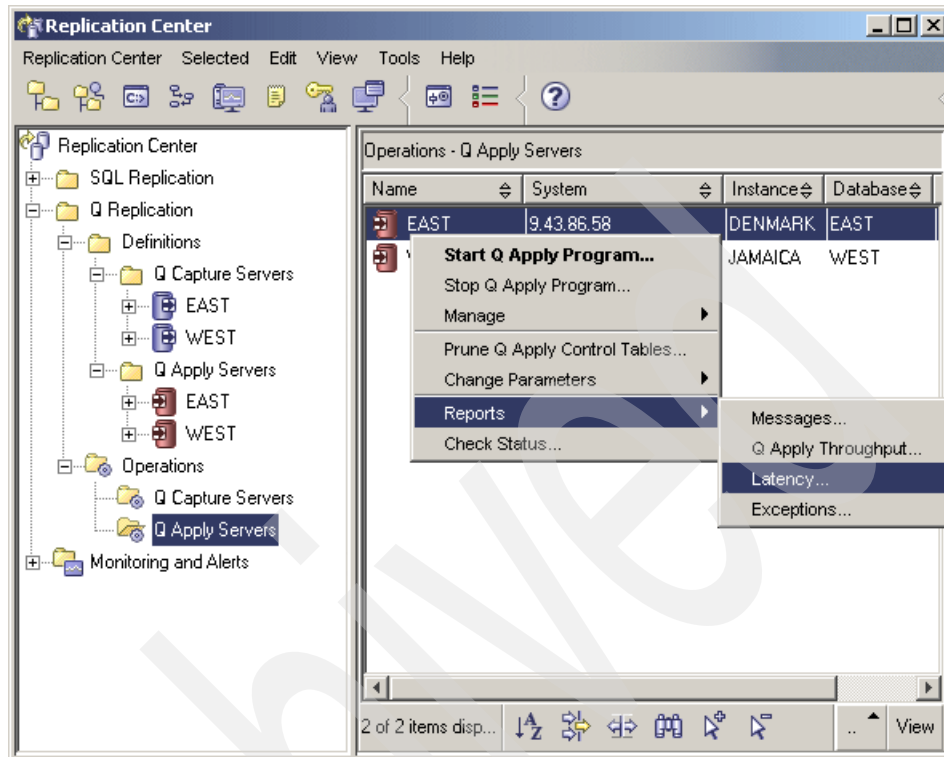


Figure 3-9 Q Capture latency and end-to-end latency using Replication Center 1/4

Latency

9.43.86.58 - DENMARK - EAST

Q Apply schema: EAST

Information to display: Composite latency

Receive queue: All receive queues

Range of time

From:

- ☐ Use first available monitor time
- ☒ Specify date and time

Date: February 22, 2006

Time: 19:25:06

To:

- ☒ Use current date and time
- ☐ Specify date and time

Date: February 23, 2006

Time: 19:25:06

Aggregate data by time intervals

Time intervals: No time interval

Rate: No time interval

Close Show SQL Show Report Help

Figure 3-10 Q Capture latency and end-to-end latency using Replication Center 2/4

Show SQL

Run System - 9.43.86.58

```
SELECT MONITOR_TIME, RECVQ, END2END_LATENCY, QLATENCY, APPLY_LATENCY,
(END2END_LATENCY-QLATENCY-APPLY_LATENCY) FROM EAST.IBMQREP_APPLYMON
WHERE MONITOR_TIME >= '2006-02-22-19.25.06.000000' ORDER BY MONITOR_TIME DESC ;
```

Close

Figure 3-11 Q Capture latency and end-to-end latency using Replication Center 3/4

| Monitor Time | Receive Queue | End-to-End Latency | Queue Latency | Q Apply Latency |
|----------------------------|---------------|--------------------|---------------|-----------------|
| 2006-02-28 11:44:46.127025 | RCVQ_W2E | 3,806 | 438 | |
| 2006-02-28 11:44:16.126685 | RCVQ_W2E | 3,549 | 444 | |
| 2006-02-28 11:43:46.126332 | RCVQ_W2E | 3,751 | 440 | |
| 2006-02-28 11:43:16.126132 | RCVQ_W2E | 3,667 | 466 | |
| 2006-02-28 11:42:46.126035 | RCVQ_W2E | 3,654 | 458 | |
| 2006-02-28 11:42:16.125607 | RCVQ_W2E | 3,666 | 436 | |
| 2006-02-28 11:41:46.125276 | RCVQ_W2E | 3,581 | 440 | |
| 2006-02-28 11:41:16.124874 | RCVQ_W2E | 3,717 | 437 | |
| 2006-02-28 11:40:46.123878 | RCVQ_W2E | 3,627 | 443 | |
| 2006-02-28 11:40:16.123459 | RCVQ_W2E | 3,645 | 437 | |
| 2006-02-28 11:39:46.123110 | RCVQ_W2E | 3,600 | 441 | |
| 2006-02-28 11:39:16.122772 | RCVQ_W2E | 3,694 | 440 | |
| 2006-02-28 11:38:46.122744 | RCVQ_W2E | 3,685 | 442 | |
| 2006-02-28 11:38:16.122467 | RCVQ_W2E | 3,631 | 438 | |
| 2006-02-28 11:37:46.121997 | RCVQ_W2E | 3,582 | 441 | |
| 2006-02-28 11:37:16.121637 | RCVQ_W2E | 3,755 | 443 | |
| 2006-02-28 11:36:46.121275 | RCVQ_W2E | 3,618 | 456 | |
| 2006-02-28 11:36:16.121175 | RCVQ_W2E | 3,801 | 436 | |
| 2006-02-28 11:35:46.120807 | RCVQ_W2E | 3,601 | 439 | |
| 2006-02-28 11:35:16.120551 | RCVQ_W2E | 3,870 | 431 | |
| 2006-02-28 11:34:46.120201 | RCVQ_W2E | 3,739 | 428 | |
| 2006-02-28 11:34:16.120113 | RCVQ_W2E | 3,622 | 428 | |
| 2006-02-28 11:33:46.119725 | RCVQ_W2E | 3,739 | 434 | |
| 2006-02-28 11:33:16.119512 | RCVQ_W2E | 3,690 | 434 | |
| 2006-02-28 11:32:46.119038 | RCVQ_W2E | 3,648 | 428 | |
| 2006-02-28 11:32:16.118629 | RCVQ_W2E | 3,698 | 428 | |
| 2006-02-28 11:31:46.118500 | RCVQ_W2E | 3,655 | 434 | |

668 of 668 items displayed

Default View View

Close

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

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 redbook.

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 3-20 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 52.

Example 3-20 Autostop Q Capture on the secondary server JAMAICA

```
asnqcmd capture_server=WEST capture_schema=WEST chgparms autostop=Y
```

```
2006-02-23-19.48.24.195923 ASN0523I "AsnQCmd" : "WEST" : "Initial" : The CHGPparms command response: "AUTOSTOP" has been set to "Y".
```

```
----Contents of stdout
```

```
2006-02-23-19.03.53.760069 ASN7000I "Q Capture" : "WEST" : "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.
```

```
2006-02-23-19.03.53.767426 ASN0572I "Q Capture" : "WEST" : "WorkerThread" : The program
initialized successfully.
```

```
2006-02-23-19.48.33.050316 ASN0573I "Q Capture" : "WEST" : "Initial" : The program was
stopped.
```

```
[asnqwk] Leaving the asnqwk thread:
```

```
[asnqwk] total database transactions captured = 1927
```

```
[asnqwk] total MQSeries transactions = 816
```

```
[asnqwk] total DB transactions published = 1927
```

```
[asnqwk] total time WAITING for logrd = 2657302 seconds
```

```
[asntxrd] Leaving logtxrd reader thread:
```

```
[asntxrd] 0 message(s) in notification queue
```

```
[asntxrd] transmgr memory in use = 0 bytes
```

```
[asntxrd] totalLogReadTime = 1.013 second(s)
```

```
[asntxrd] total logrd SLEEP time = 2675.006 second(s)
```

```
[asntxrd] totalLogRecsRead = 3861
```

```
[asntxrd] Queued a thread Termination Notification.
```

```
[waitForLogrdEnd] The logrd thread terminated with rc: 0
```

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 CURDEPTH field in Example 3-21 on page 66 is zero indicating that the transmit queue XMIT_W2E on the secondary server has been fully consumed.

Example 3-21 Verify XMITQ on the secondary server JAMAICA is fully consumed

```
echo "dis ql(XMIT_W2E) curdepth " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : dis ql(XMIT_W2E) curdepth
AMQ8409: Display Queue details.
      QUEUE(XMIT_W2E)                                TYPE(QLLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

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 recommend 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 CURDEPTH field in Example 3-22 is zero indicating that the receive queue RCVQ_W2E on the primary server has been fully consumed.

Example 3-22 Verify RECVQ on the primary server DENMARK is fully consumed

```
echo "dis ql(RCVQ_W2E) curdepth " | runmqsc QM_EAST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : dis ql(RCVQ_W2E) curdepth
AMQ8409: Display Queue details.
      QUEUE(RCVQ_W2E)                                TYPE(QLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

STEP SWCF11: Resume workload on primary

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 redbook.

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 59.

STEP SWCF13: Ensure no exceptions recorded

This is a precautionary step since no exceptions should be occur on a switchback from a controlled failover. Figure 3-13 on page 68 through Figure 3-16 on page 70 show the Replication Center GUI for viewing exceptions. Even though Figure 3-14 on page 69 shows only “Unexpected SQL errors” being selected, as a precaution all the categories should be selected.

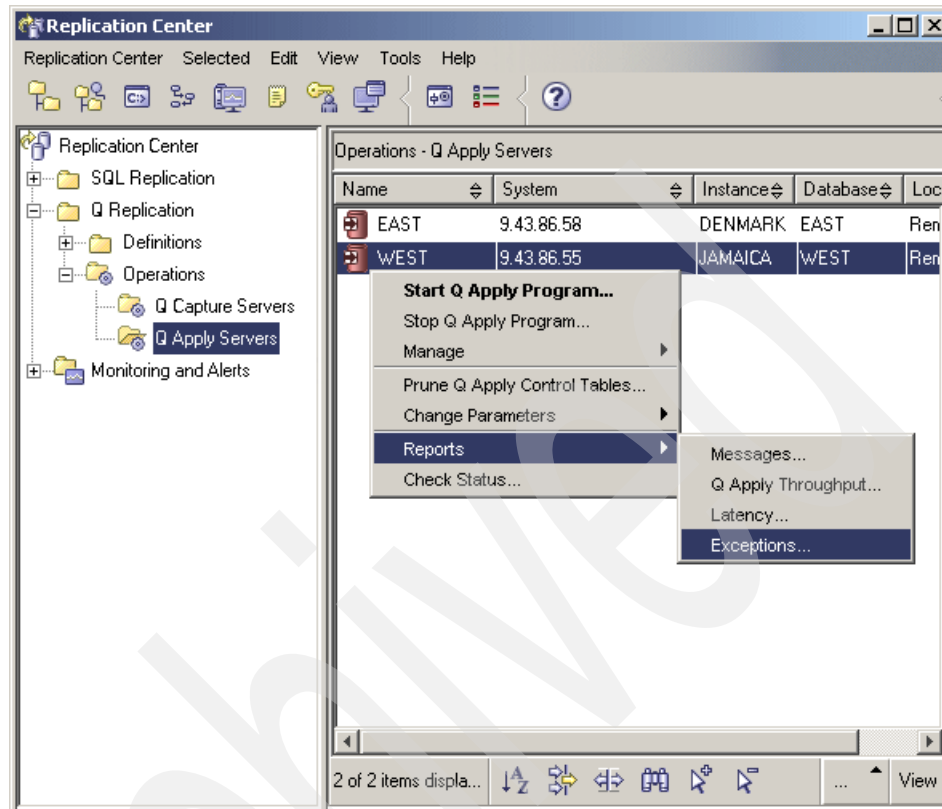


Figure 3-13 Viewing exceptions through Replication Center 1/4

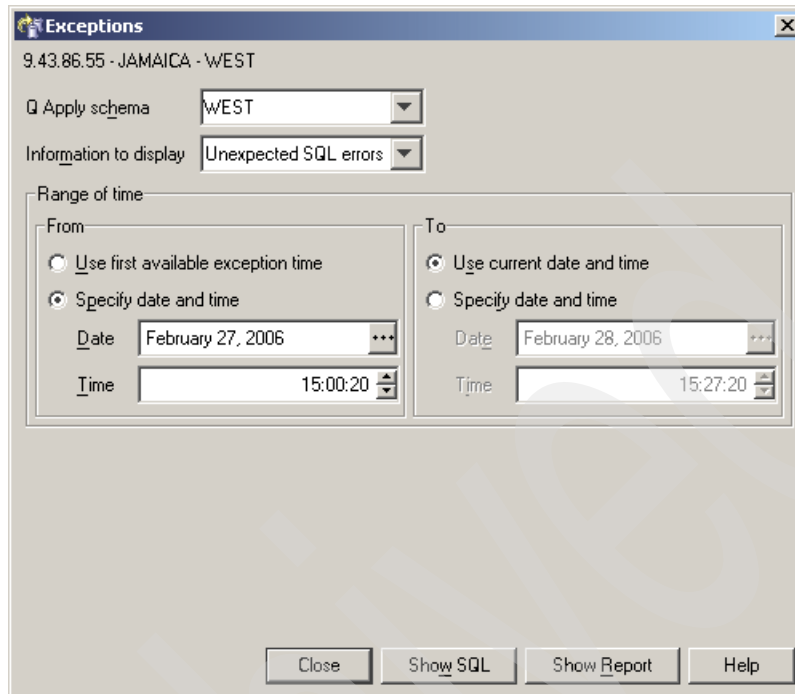


Figure 3-14 Viewing exceptions through Replication Center 2/4

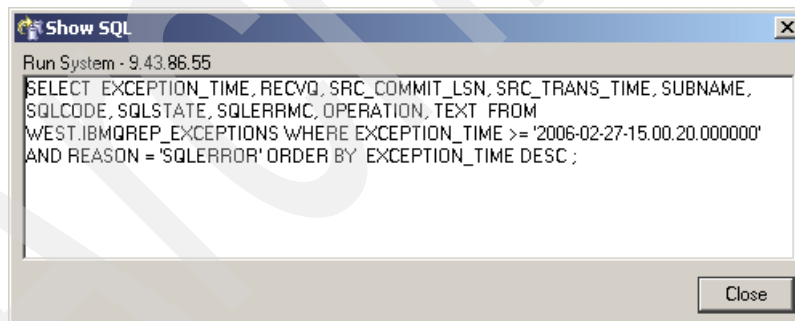


Figure 3-15 Viewing exceptions through Replication Center 3/4

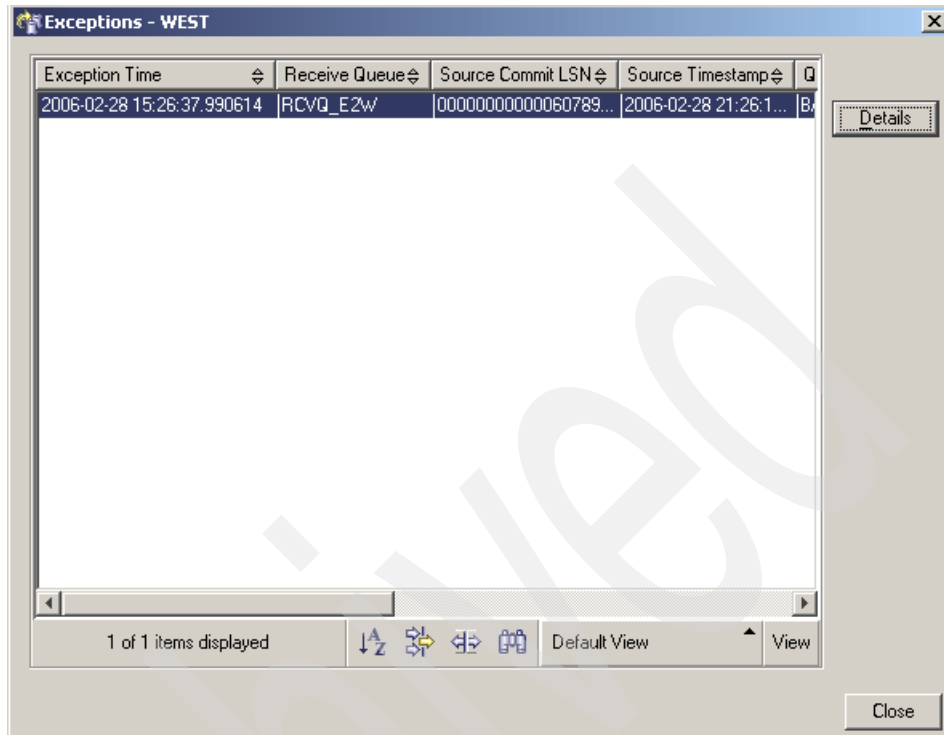


Figure 3-16 Viewing exceptions through Replication Center 4/4

3.7 Uncontrolled failover (UF) and switchback

The uncontrolled failover scenario is driven by circumstances that force and 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 225.
 - 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.

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

Important (continued):

- 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.

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 3-17 on page 73 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.

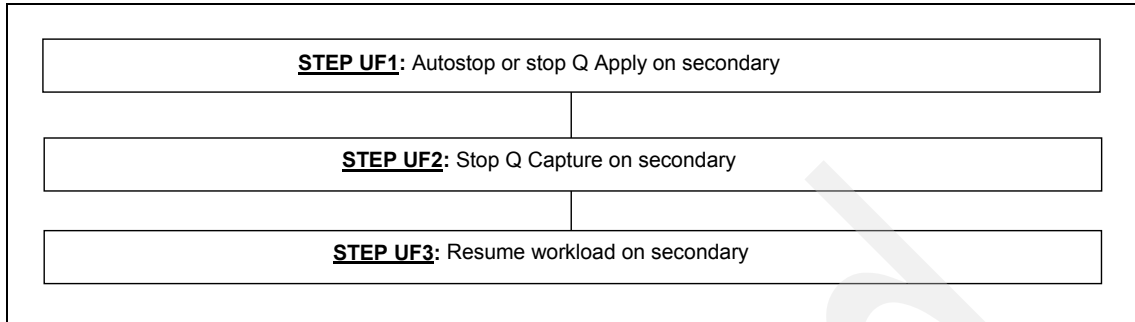


Figure 3-17 Overview of uncontrolled failover (UF) steps

The steps shown in Figure 3-17 are described briefly in the following sections.

STEP UF1: Autostop 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 55.

Attention: If Q Apply does not shutdown, it could be because it 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 74.

If the `CURDEPTH` field remains not at zero for a while, 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 75.

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 asnqfmt.
--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

Example 3-24 on page 75 shows the WebSphere MQ command to determine that the receive queue is empty. The value of the CURDEPTH field is zero indicating that the receive queue RCVQ_E2W on the secondary server has been fully consumed.

Example 3-24 Verify RECVQ on the secondary server JAMAICA is fully consumed

```
echo "dis ql(RCVQ_E2W) curdepth " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : dis ql(RCVQ_E2W) curdepth
AMQ8409: Display Queue details.
      QUEUE(RCVQ_E2W)                                TYPE(QLLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

STEP UF1b: Stop Q Apply on secondary

Example 3-25 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 similar to that shown in Example 3-11 on page 55. You may also monitor the Q Apply process by issuing the `ps -ef | grep asnqapp | grep -v grep` command.

Example 3-25 Stop Q Apply on the secondary server JAMAICA

```
asnqacmd apply_server=WEST apply_schema=WEST STOP
```

```
2006-02-24-17.35.40.869747 ASN0522I "AsnQAcmd" : "WEST" : "Initial" : The program received
the "STOP" command.
```

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 52.

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 redbook.

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 46 summarizes the considerations in 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 3-18 on page 77 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.
- ▶ Archive log files on the secondary server are kept on DB2 ARCHIVE LOGPATH, and are not removed during the outage of the primary server.
- ▶ 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.

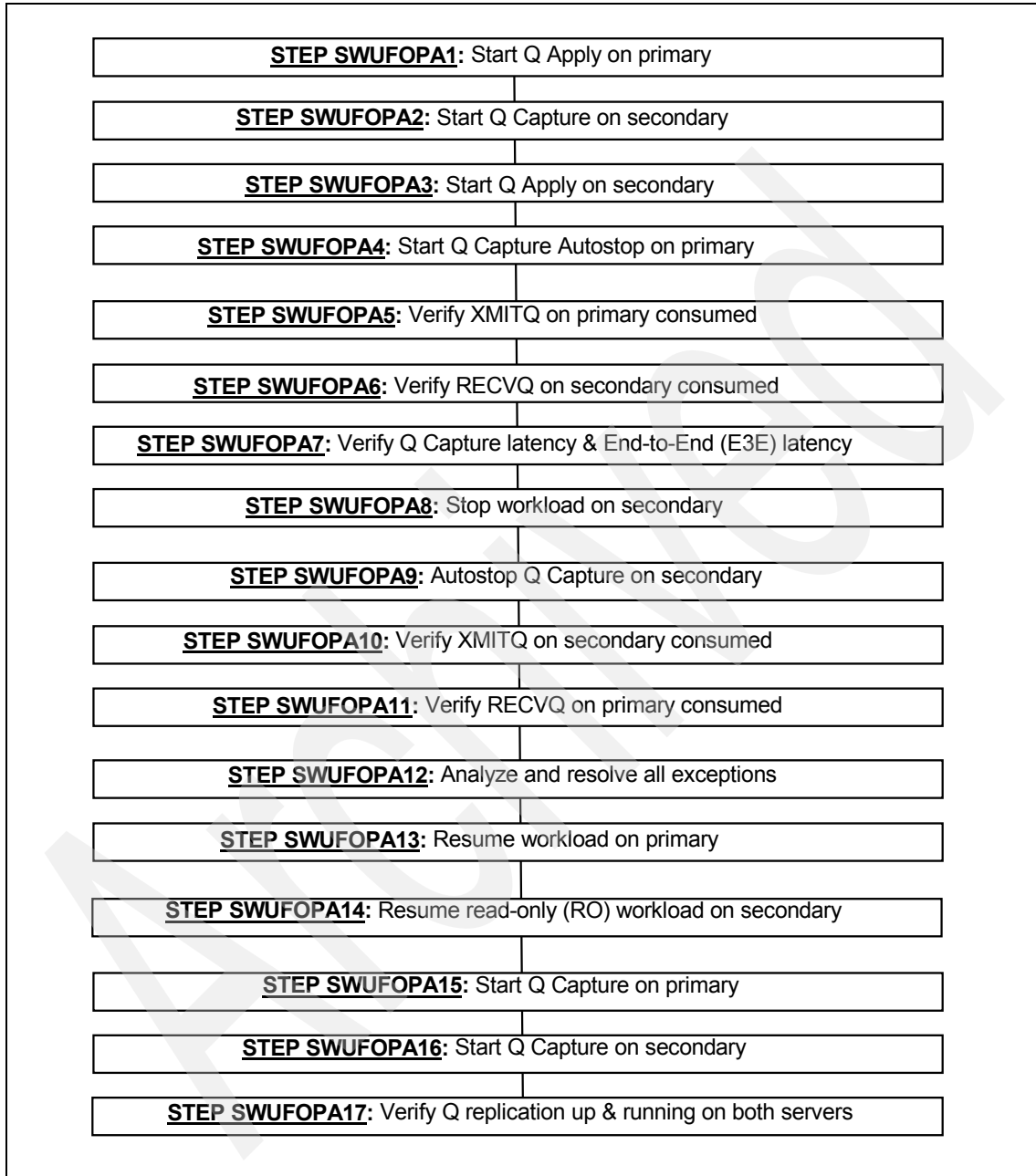


Figure 3-18 Overview of Option A steps

The steps shown in Figure 3-18 are 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 57.

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 59.

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 59.

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 3-15 on page 58. You may also monitor the Q Capture process by issuing the **ps -ef | grep asnqcap | grep -v grep** command.

Example 3-26 Autostop Q Capture on the primary server DENMARK

```
asnqcap capture_server=EAST capture_schema=EAST capture_path=/DB2 AUTOSTOP=Y
```

```
2006-02-24-19.00.03.507288 ASN7000I  "Q Capture" : "EAST" : "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.
```

```
2006-02-24-19.00.03.512847 ASN0572I  "Q Capture" : "EAST" : "WorkerThread" : The program
initialized successfully.
```

```
2006-02-24-19.00.12.444039 ASN0573I  "Q Capture" : "EAST" : "Initial" : The program was
stopped.
```

----Contents of stdout

```
[asnqwk] Leaving the asnqwk thread:
[asnqwk] total database transactions captured = 2
[asnqwk] total MQSeries transactions          = 1
[asnqwk] total DB transactions published      = 2
[asnqwk] total time WAITING for logrd        = 2 seconds
[asntxrd] Leaving logtxrd reader thread:
[asntxrd] 0 message(s) in notification queue
[asntxrd] transmgr memory in use              = 0 bytes
[asntxrd] totalLogReadTime                    = 0.003 second(s)
[asntxrd] total logrd SLEEP time              = 5.000 second(s)
[asntxrd] totalLogRecsRead                    = 6
```

[asntxrd] Queued a thread Termination Notification.
[waitForLogrdEnd] The logrd thread terminated with rc: 0

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 54.

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 74.

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 have been replicated over to the primary server as described in “STEP SWCF5: Verify Q Capture latency and E2E latency” on page 59.

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 redbook.

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 65.

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 65.

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 66.

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 225.

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 workload on primary

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 redbook.

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 58.

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 59.

STEP SWUFOPA17: Verify 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 on page 81 through Example 3-30 on page 81.
- ▶ By checking for active Q replication processes, and active subscription states on the primary and secondary servers as shown in Example 3-31 on page 82 and Example 3-32 on page 82 respectively.

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

asnqacmd apply_server=EAST apply_schema=EAST status

2006-02-24-19.16.43.634031 ASN0520I "AsnQAcmd" : "EAST" : "Initial" : The STATUS command response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-19.16.43.634203 ASN0520I "AsnQAcmd" : "EAST" : "Initial" : The STATUS command response: "AdminThread" thread is in the "is resting" state.
2006-02-24-19.16.43.634235 ASN0520I "AsnQAcmd" : "EAST" : "Initial" : The STATUS command response: "MonitorThread" thread is in the "is resting" state.
2006-02-24-19.16.43.634259 ASN0520I "AsnQAcmd" : "EAST" : "Initial" : The STATUS command response: "BR00000" thread is in the "is doing work" state.

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

asnqacmd apply_server=WEST apply_schema=WEST status

2006-02-24-19.17.20.303912 ASN0520I "AsnQAcmd" : "WEST" : "Initial" : The STATUS command response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-19.17.20.304016 ASN0520I "AsnQAcmd" : "WEST" : "Initial" : The STATUS command response: "AdminThread" thread is in the "is resting" state.
2006-02-24-19.17.20.304039 ASN0520I "AsnQAcmd" : "WEST" : "Initial" : The STATUS command response: "MonitorThread" thread is in the "is resting" state.
2006-02-24-19.17.20.304061 ASN0520I "AsnQAcmd" : "WEST" : "Initial" : The STATUS command response: "BR00000" thread is in the "is doing work" state.

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

asnqccmd capture_server=EAST capture_schema=EAST status

2006-02-24-19.18.52.455681 ASN0520I "AsnQCcmd" : "EAST" : "Initial" : The STATUS command response: "HoldLThread" thread is in the "is waiting" state.
2006-02-24-19.18.52.455853 ASN0520I "AsnQCcmd" : "EAST" : "Initial" : The STATUS command response: "AdminThread" thread is in the "is resting" state.
2006-02-24-19.18.52.455885 ASN0520I "AsnQCcmd" : "EAST" : "Initial" : The STATUS command response: "PruneThread" thread is in the "is resting" state.
2006-02-24-19.18.52.455909 ASN0520I "AsnQCcmd" : "EAST" : "Initial" : The STATUS command response: "WorkerThread" thread is in the "is doing work" state.

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

asnqccmd capture_server=WEST capture_schema=WEST status

2006-02-24-19.19.39.781430 ASN0520I "AsnQCcmd" : "WEST" : "Initial" : The STATUS command response: "HoldLThread" thread is in the "is waiting" state.

```

2006-02-24-19.19.39.781537 ASN0520I "AsnQCcmd" : "WEST" : "Initial" : The
STATUS command response: "AdminThread" thread is in the "is resting" state.
2006-02-24-19.19.39.781561 ASN0520I "AsnQCcmd" : "WEST" : "Initial" : The
STATUS command response: "PruneThread" thread is in the "is resting" state.
2006-02-24-19.19.39.781582 ASN0520I "AsnQCcmd" : "WEST" : "Initial" : The
STATUS command response: "WorkerThread" thread is in the "is doing work" state.

```

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

```
ps -ef | grep asn | grep -v grep
```

```

-----primary server denmark-----
qrepladm 753816      1  0 19:09:28 pts/6  0:00 asnqcap capture_server=EAST
capture_schema=EAST capture_path=/DB2
qrepladm 794780      1  0 18:47:23 pts/6  0:00 asnqapp apply_server=EAST
apply_schema=EAST apply_path=/DB2

-----secondary server jamaica-----
qrepladm 1032286     1  0 19:11:10 pts/5   0:00 asnqcap capture_server=WEST
capture_schema=WEST capture_path=/DB2
qrepladm 1081426     1  0 18:56:28 pts/5   0:00 asnqapp apply_server=WEST
apply_schema=WEST apply_path=/DB2

```

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

```
-----primary server denmark-----
```

```
connect to EAST
```

```
Database Connection Information
```

```

Database server      = DB2/AIX64 9.1.0
SQL authorization ID = QREPLADM
Local database alias = EAST

```

```
select substr(repqmapname,1,15) as repqmapname, state from
EAST.ibmqrep_recvqueues
```

```

REPQMAPNAME      STATE
-----
W_TO_E_MAP       A

```

```
1 record(s) selected.
```

```
select substr(subname,1,15) as subname,state,before_values,
changed_cols_only,has_loadphase from EAST.ibmqrep_subs
```

| SUBNAME | STATE | BEFORE_VALUES | CHANGED_COLS_ONLY | HAS_LOADPHASE |
|-------------|-------|---------------|-------------------|---------------|
| BAL0001 | A | Y | Y | I |
| OPTIONS0001 | A | Y | Y | I |
| PROD0001 | A | Y | Y | I |
| TRAN0001 | A | Y | Y | I |

4 record(s) selected.

```
select substr(subname,1,15) as subname,state,conflict_action,
conflict_rule,has_loadphase from EAST.ibmqrep_targets
```

| SUBNAME | STATE | CONFLICT_ACTION | CONFLICT_RULE | HAS_LOADPHASE |
|-------------|-------|-----------------|---------------|---------------|
| BAL0002 | A | F | C | I |
| PROD0002 | A | F | C | I |
| TRAN0002 | A | F | C | I |
| OPTIONS0002 | A | F | C | I |

4 record(s) selected.

connect reset

DB20000I The SQL command completed successfully.

-----secondary server jamaica-----

connect to WEST

Database Connection Information

Database server = DB2/AIX64 9.1.0

SQL authorization ID = QREPLADM

Local database alias = WEST

```
select substr(repqmapname,1,15) as repqmapname, state from
WEST.ibmqrep_rcvqueues
```

| REPQMAPNAME | STATE |
|-------------|-------|
| E_TO_W_MAP | A |

1 record(s) selected.

```
select substr(subname,1,15) as subname,state,before_values,
changed_cols_only,has_loadphase from WEST.ibmqrep_subs
```

| SUBNAME | STATE | BEFORE_VALUES | CHANGED_COLS_ONLY | HAS_LOADPHASE |
|-------------|-------|---------------|-------------------|---------------|
| BAL0002 | A | Y | N | I |
| OPTIONS0002 | A | Y | N | I |
| PROD0002 | A | Y | N | I |
| TRAN0002 | A | Y | N | I |

4 record(s) selected.

```
select substr(subname,1,15) as subname,state,conflict_action,
conflict_rule,has_loadphase from WEST.ibmqrep_targets
```

| SUBNAME | STATE | CONFLICT_ACTION | CONFLICT_RULE | HAS_LOADPHASE |
|-------------|-------|-----------------|---------------|---------------|
| BAL0001 | A | I | C | I |
| OPTIONS0001 | A | I | C | I |
| PROD0001 | A | I | C | I |
| TRAN0001 | A | I | C | I |

4 record(s) selected.

```
connect reset
```

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

3.7.3 Option B: Resync primary and reinitialize secondary

Table 3-1 on page 46 summarizes the considerations in 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 subscription(s) is inactivated 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 3-19 on page 86 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.
- ▶ Archive log files on the secondary server are kept on DB2 ARCHIVE LOGPATH, and are not removed during the outage of the primary server.
- ▶ 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.

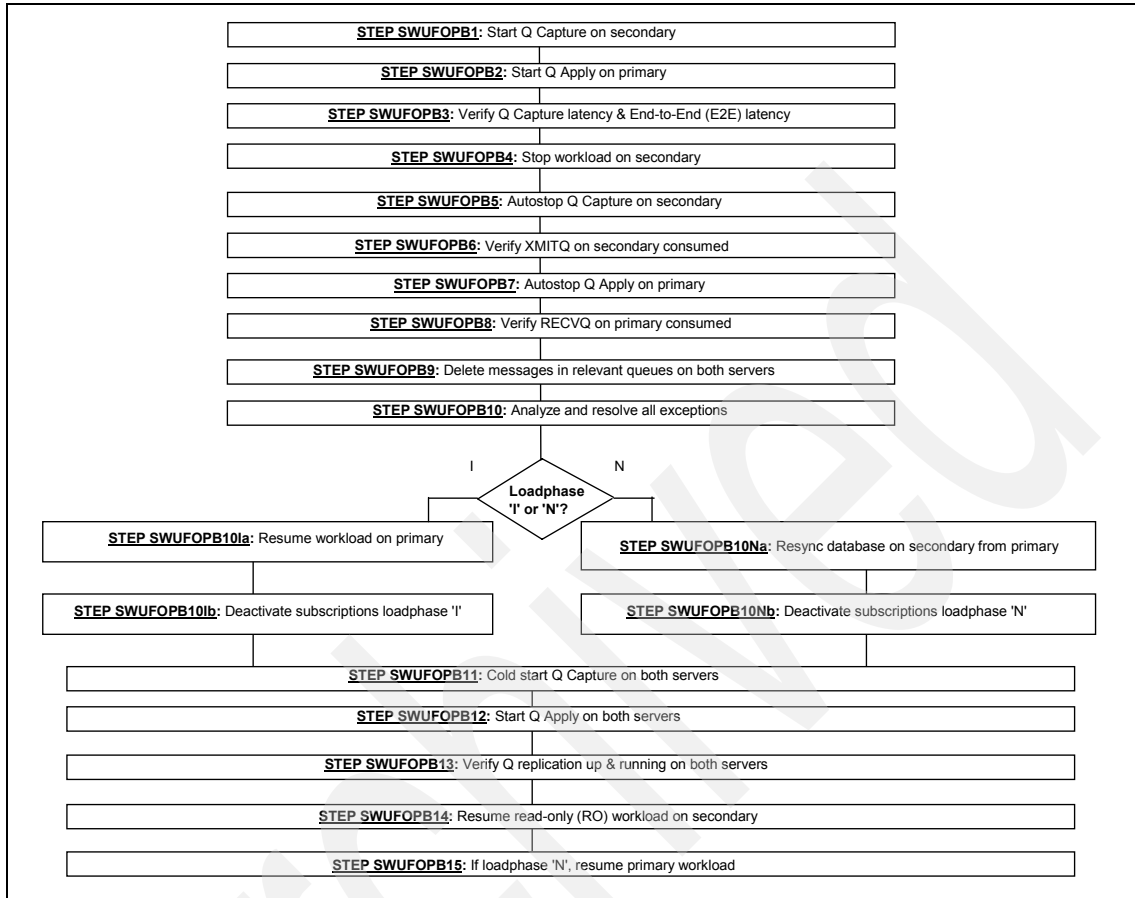


Figure 3-19 Overview of Option B steps

The steps shown in Figure 3-19 are 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 59.

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 57.

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 have been replicated over to the primary server. "STEP SWCF5: Verify Q Capture latency and E2E latency" on page 59 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 redbook.

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 65.

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 65.

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 on page 88. You may also monitor the Q Apply process by issuing the `ps -ef | grep asnqapp | grep -v grep` command.

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

```
asnqacmd apply_server=EAST apply_schema=EAST chgparms autostop=Y
```

```
2006-02-28-10.26.09.521428 ASN0523I "AsnQAcmd" : "EAST" : "Initial" : The CHGPparms command  
response: "AUTOSTOP" has been set to "Y".
```

All valid MQSC commands were processed.

echo "DIS CHSTATUS(E2W)" | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : DIS CHSTATUS(E2W)
AMQ8417: Display Channel Status details.
CHANNEL(E2W)                CHLTYPE(SDR)
CONNNAME(9.43.86.55(1450))   CURRENT
RQMNAME(QM_WEST)             STATUS(STOPPED)
SUBSTATE( )                  XMITQ(XMIT_E2W)

One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

echo "CLEAR QLOCAL(XMIT_E2W) " | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : CLEAR QLOCAL(XMIT_E2W)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

echo "DIS QL(XMIT_E2W) CURDEPTH " | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : DIS QL(XMIT_E2W) CURDEPTH
AMQ8409: Display Queue details.
QUEUE(XMIT_E2W)              TYPE(QLOCAL)
CURDEPTH(0)

One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

echo "START CHL(E2W) " | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```

1 : START CHL(E2W)
AMQ8018: Start WebSphere MQ channel accepted.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
-----
echo "DIS CHSTATUS(E2W)" | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.


1 : DIS CHSTATUS(E2W)
AMQ8417: Display Channel Status details.
CHANNEL(E2W)                                CHLTYPE(SDR)
CONNNAME(9.43.86.55(1450))                  CURRENT
RQMNAME(QM_WEST)                            STATUS(RUNNING)
SUBSTATE(MQGET)                             XMITQ(XMIT_E2W)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
-----
echo "CLEAR QLOCAL(RCVQ_W2E) " | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.


1 : CLEAR QLOCAL(RCVQ_W2E)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
-----
echo "DIS QL(RCVQ_W2E) CURDEPTH" | runmqsc QM_EAST

5724-H72 (C) Copyright IBM Corp. 1994, 2005.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.


1 : DIS QL(RCVQ_W2E) CURDEPTH
AMQ8409: Display Queue details.
QUEUE(RCVQ_W2E)                                TYPE(QLOCAL)
CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
-----

```

```
echo "CLEAR QLOCAL(EAST_ADMINQ)" | runmqsc QM_EAST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : CLEAR QLOCAL(EAST_ADMINQ)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
-----
echo "DIS QL(EAST_ADMINQ) CURDEPTH" | runmqsc QM_EAST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : DIS QL(EAST_ADMINQ) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(EAST_ADMINQ)                TYPE(QLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
-----
echo "CLEAR QLOCAL(EAST_RESTARTQ) " | runmqsc QM_EAST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : CLEAR QLOCAL(EAST_RESTARTQ)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
-----
echo "DIS QL(EAST_RESTARTQ) CURDEPTH" | runmqsc QM_EAST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_EAST.

```
1 : DIS QL(EAST_RESTARTQ) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(EAST_RESTARTQ)              TYPE(QLOCAL)
      CURDEPTH(0)
One MQSC command read.
```

No commands have a syntax error.
All valid MQSC commands were processed.

Example 3-36 Delete messages in XMITQ, RECVQ, ADMINQ, RESTARTQ on the secondary server JAMAICA

```
echo "STOP CHL(W2E) " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : STOP CHL(W2E)
AMQ8019: Stop WebSphere MQ channel accepted.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
echo "DIS CHSTATUS(W2E)" | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : DIS CHSTATUS(W2E)
AMQ8417: Display Channel Status details.
CHANNEL(W2E)                CHLTYPE(SDR)
CONNNAME(9.43.86.58(1451))   CURRENT
RQMNAME(QM_EAST)             STATUS(STOPPED)
SUBSTATE( )                  XMITQ(XMIT_W2E)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
echo "CLEAR QLOCAL(XMIT_W2E) " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : CLEAR QLOCAL(XMIT_W2E)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
echo "DIS QL(XMIT_W2E) CURDEPTH " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.

Starting MQSC for queue manager QM_WEST.

```
1 : DIS QL(XMIT_W2E) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(XMIT_W2E)                                TYPE(QLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

echo "START CHL(W2E) " | runmqsc QM_WEST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : START CHL(W2E)
AMQ8018: Start WebSphere MQ channel accepted.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

echo "DIS CHSTATUS(W2E)" | runmqsc QM_WEST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : DIS CHSTATUS(W2E)
AMQ8417: Display Channel Status details.
      CHANNEL(W2E)                                CHLTYPE(SDR)
      CONNAME(9.43.86.58(1451))                    CURRENT
      RQMNAME(QM_EAST)                             STATUS(RUNNING)
      SUBSTATE(MQGET)                             XMITQ(XMIT_W2E)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

echo "CLEAR QLOCAL(RCVQ_E2W) " | runmqsc QM_WEST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : CLEAR QLOCAL(RCVQ_E2W)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
```

All valid MQSC commands were processed.

```
echo "DIS QL(RCVQ_E2W) CURDEPTH" | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : DIS QL(RCVQ_E2W) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(RCVQ_E2W)                                TYPE(QLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
echo "CLEAR QLOCAL(WEST_ADMINQ) " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : CLEAR QLOCAL(WEST_ADMINQ)
AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
echo "DIS QL(WEST_ADMINQ) CURDEPTH" | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : DIS QL(WEST_ADMINQ) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(WEST_ADMINQ)                                TYPE(QLOCAL)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
echo "CLEAR QLOCAL(WEST_RESTARTQ) " | runmqsc QM_WEST
```

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : CLEAR QLOCAL(WEST_RESTARTQ)
```

AMQ8022: WebSphere MQ queue cleared.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.

echo "DIS QL(WEST_RESTARTQ) CURDEPTH" | runmqsc QM_WEST

5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM_WEST.

```
1 : DIS QL(WEST_RESTARTQ) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(WEST_RESTARTQ)                TYPE(QLocal)
      CURDEPTH(0)
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

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 225.

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 workload on primary” on page 96 and “STEP SWUFOPB10Ib: Deactivate subscriptions loadphase 'I'” on page 96 before executing “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 100.
- ▶ If the NO load option is chosen, follow steps “STEP SWUFOPB10Na: Resync database on secondary from primary” on page 98 and “STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'” on page 98 before executing “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 100.

STEP SWUFOPB10la: Resume workload on primary

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 redbook.

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 SWUFOPB10lb: 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 DENMARK

CONNECT TO EAST USER db2inst9 USING

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = EAST

UPDATE EAST.IBMQREP_SUBS SET state='N', sub_id=NULL, group_members=NULL, state_transition=NULL, has_loadphase = 'I'

DB20000I The SQL command completed successfully.

UPDATE EAST.IBMQREP_TARGETS SET state='I', sub_id=NULL

DB20000I The SQL command completed successfully.

UPDATE EAST.IBMQREP_RECVQUEUES SET state='A'

DB20000I The SQL command completed successfully.

UPDATE EAST.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 JAMAICA

CONNECT TO WEST USER db2inst9 USING

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = WEST

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

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

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

```
UPDATE WEST.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

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 redbook.

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 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 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 3-38 on page 97 shows the SQL for deactivating subscriptions on the primary server, while Example 3-39 shows the same for the secondary server.

Example 3-39 Deactivate subscriptions on the primary server DENMARK

CONNECT TO EAST USER db2inst9 USING

Database Connection Information

Database server = DB2/AIX64 9.1.0
 SQL authorization ID = DB2INST9
 Local database alias = EAST

UPDATE EAST.IBMQREP_SUBS SET state='N', sub_id=NULL, group_members=NULL, state_transition=NULL, has_loadphase='N'

DB20000I The SQL command completed successfully.

UPDATE EAST.IBMQREP_TARGETS SET state='I', sub_id=NULL

DB20000I The SQL command completed successfully.

UPDATE EAST.IBMQREP_RECVQUEUES SET state='A'

DB20000I The SQL command completed successfully.

UPDATE EAST.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 JAMAICA

CONNECT TO WEST USER db2inst9 USING

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = WEST

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

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

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

```
UPDATE WEST.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, start 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 on page 101, the output of which indicates that the Q Capture program initialized successfully. You can monitor the contents of the Q Capture log which shows the Q Capture program initializing successfully as shown in Example 3-42 on page 101. You may also monitor the Q Capture process by issuing the `ps -ef | grep asnqcap | grep -v grep` command.
- ▶ Cold start Q Capture on the secondary server by submitting the job shown in Example 3-43 on page 101, the output of which indicates that the Q Capture program initialized successfully. You can monitor the contents of the Q Capture log which shows the Q Capture program initializing successfully as shown in Example 3-44 on page 101. You may also monitor the Q Capture process by issuing the `ps -ef | grep asnqcap | grep -v grep` command.

Example 3-41 Cold start Q Capture on the primary server DENMARK

asnqcap capture_server=EAST capture_schema=EAST capture_path=/DB2 startmode=cold

2006-02-28-10.32.40.863717 ASN7000I "Q Capture" : "EAST" : "WorkerThread" : "0"
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.

**2006-02-28-10.32.40.894977 ASN0572I "Q Capture" : "EAST" : "WorkerThread" : The program
initialized successfully.**

Example 3-42 Cold start Q Capture log contents on the primary server DENMARK

.....

**2006-02-28-10.32.39.704505 <asnParmClass::printParms> ASN0529I "Q Capture" :
"EAST" : "Initial" : The value of "STARTMODE" was set to "COLD" at startup by
the following method: "COMMANDLINE".**

.....

2006-02-28-10.32.40.863717 <subMgr::processSubscriptions> ASN7000I "Q Capture"
: "EAST" : "WorkerThread" : "0" 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.

2006-02-28-10.32.40.881557 <waitForLogrdInit> ASN7108I "Q Capture" : "EAST" :
"WorkerThread" : At program initialization, the highest log sequence number of
a successfully processed transaction is "4404:7B27:0000:0000:0000" and the
lowest log sequence number of a transaction still to be committed is
"0000:0000:0000:5E7F:94BB".

**2006-02-28-10.32.40.894977 <asnqwk> ASN0572I "Q Capture" : "EAST" :
"WorkerThread" : The program initialized successfully.**

Example 3-43 Cold start Q Capture on the secondary server JAMAICA

asnqcap capture_server=WEST capture_schema=WEST capture_path=/DB2 startmode=cold

2006-02-28-10.34.28.790706 ASN7000I "Q Capture" : "WEST" : "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.

**2006-02-28-10.34.28.811825 ASN0572I "Q Capture" : "WEST" : "WorkerThread" : The program
initialized successfully.**

Example 3-44 Cold start Q Capture log contents on the secondary server JAMAICA

.....

**2006-02-28-10.34.27.747888 <asnParmClass::printParms> ASN0529I "Q Capture" :
"WEST" : "Initial" : The value of "STARTMODE" was set to "COLD" at startup by
the following method: "COMMANDLINE".**

.....

```
2006-02-28-10.34.28.790706 <subMgr::processSubscriptions> ASN7000I "Q Capture"
: "WEST" : "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.
2006-02-28-10.34.28.811771 <waitForLogrdInit> ASN7108I "Q Capture" : "WEST" :
"WorkerThread" : At program initialization, the highest log sequence number of
a successfully processed transaction is "4404:7B93:0000:0000:0000" and the
lowest log sequence number of a transaction still to be committed is
"0000:0000:0000:16C0:A749".
2006-02-28-10.34.28.811825 <asnqwk> ASN0572I "Q Capture" : "WEST" :
"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 57, and on the secondary server as described in “STEP SWCF4: Start Q Apply on secondary” on page 59.

STEP SWUFOPB13: Verify Q replication up and running

Verify that normal operations have been restored as described in “STEP SWUFOPA17: Verify Q replication up and running” on page 80.

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 redbook.

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 workload on primary” on page 96.

3.7.4 Option C: Resync secondary and reinitialize primary

Table 3-1 on page 46 summarizes the considerations in 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 subscription(s) is 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 3-20 on page 104 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.

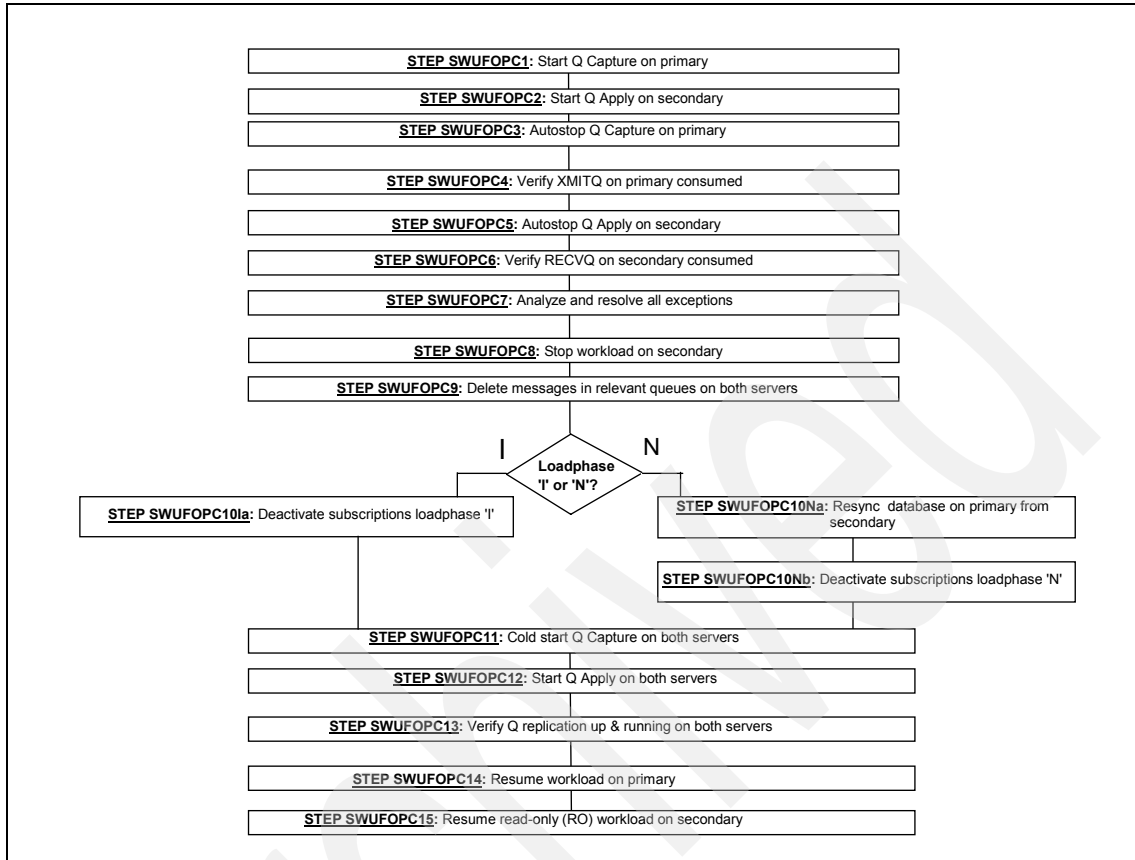


Figure 3-20 Overview of Option C steps

The steps shown in Figure 3-20 are 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 58.

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 59.

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 51.

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 54.

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 55.

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 74.

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 225.

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 redbook.

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

STEP SWUFOPC9: Delete messages in relevant queues

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 page 88.

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 steps “STEP SWUFOPC10la: Deactivate subscriptions loadphase 'I'” on page 106 followed by “STEP SWUFOPC11: Cold start Q Capture on both servers” on page 108.
- ▶ If the NO load option is chosen, follow steps “STEP SWUFOPB10Na: Resync database on secondary from primary” on page 98 and “STEP SWUFOPB10Nb: Deactivate subscriptions loadphase 'N'” on page 98 before executing “STEP SWUFOPB11: Cold start Q Capture on both servers” on page 100.

STEP SWUFOPC10la: 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 3-45 shows the SQL for deactivating subscriptions on the primary server, while Example 3-46 on page 108 shows the same for the secondary server.

Example 3-45 Deactivate subscriptions on the primary server DENMARK

CONNECT TO EAST USER db2inst9 USING

Database Connection Information

Database server = DB2/AIX64 9.1.0
 SQL authorization ID = DB2INST9
 Local database alias = EAST

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

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

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

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

CONNECT RESET

DB20000I The SQL command completed successfully.

Example 3-46 Deactivate subscriptions on the secondary server JAMAICA

CONNECT TO WEST USER db2inst9 USING

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = WEST

**UPDATE WEST.IBMQREP_SUBS SET state='N', sub_id=NULL, group_members=NULL,
state_transition=NULL, has_loadphase='I'**

DB20000I The SQL command completed successfully.

UPDATE WEST.IBMQREP_TARGETS SET state='I', sub_id=NULL

DB20000I The SQL command completed successfully.

UPDATE WEST.IBMQREP_RECVQUEUES SET state='A'

DB20000I The SQL command completed successfully.

UPDATE WEST.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 98.

STEP SWUFOPC11: 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 100.

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 57, and on the secondary server as described in “STEP SWCF4: Start Q Apply on secondary” on page 59.

STEP SWUFOPC13: Verify Q replication up and running

Verify that Q replication is up and running on both servers as described in “STEP SWUFOPA17: Verify Q replication up and running” on page 80.

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 SWUFOPC10la: Deactivate subscriptions loadphase ‘I’” on page 106.

STEP SWUFOPC14: Resume workload on primary

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 redbook.

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 46 summarizes the considerations in choosing between four possible switchback options. Option D involves discarding all unpropagated 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 is 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 114 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 3-21 on page 111 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.

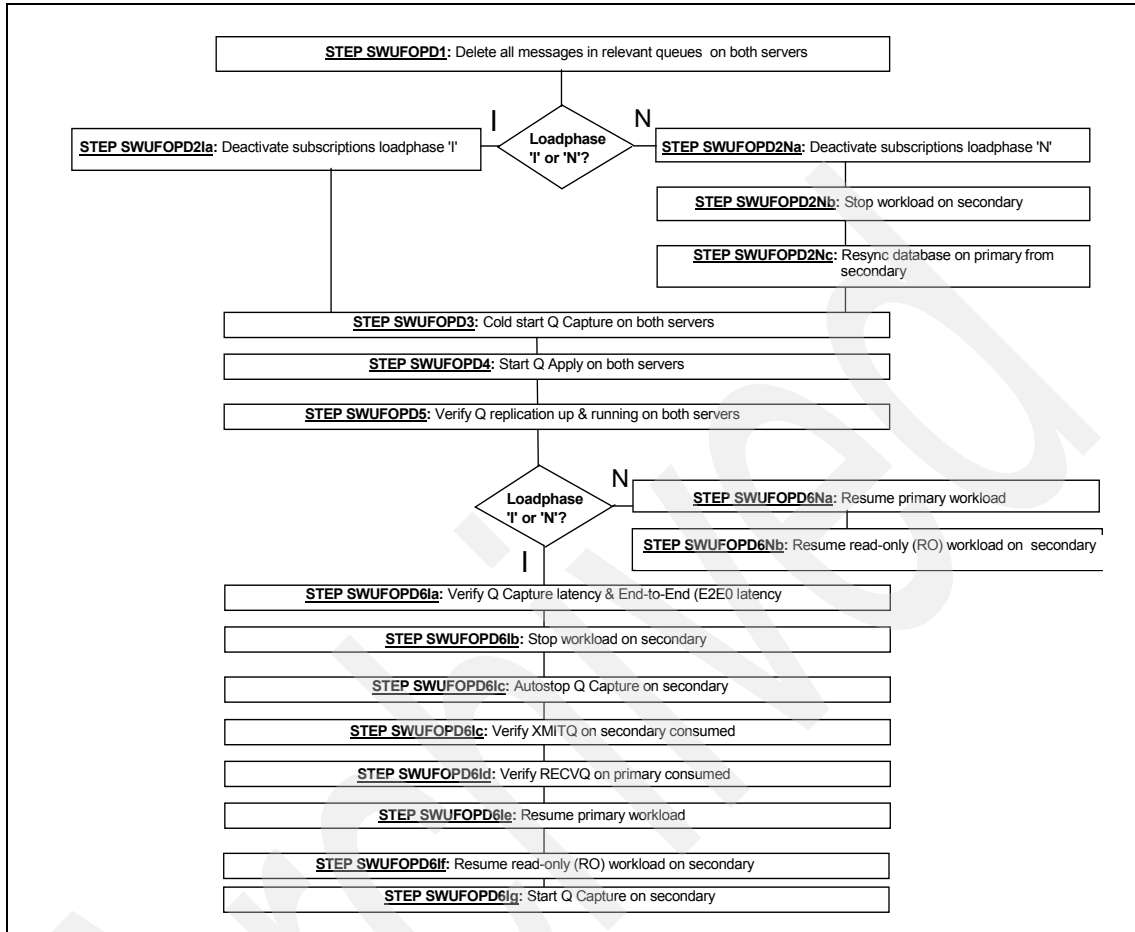


Figure 3-21 Overview of Option D steps

The steps shown in Figure 3-21 are described briefly in the following sections.

STEP SWUFOPD1: Delete all messages in relevant queues

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 page 88. 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 steps “STEP SWUFOPD2Ia: Deactivate subscriptions loadphase 'I'” on page 112 and “STEP SWUFOPB10Ib: Deactivate subscriptions loadphase 'I'” on page 96 before executing “STEP SWUFOPD3: Cold start Q Capture on both servers” on page 112.
- ▶ If the NO load option is chosen, follow steps “STEP SWUFOPD2Na: Deactivate subscriptions loadphase 'N'” on page 112, “STEP SWUFOPD2Nb: Stop workload on secondary” on page 112 and “STEP SWUFOPD2Nc: Resync database on primary from secondary” on page 112, before executing “STEP SWUFOPD3: Cold start Q Capture on both servers” on page 112.

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 106.

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 98.

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 redbook.

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

STEP SWUFOPD2Nc: 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 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 100.

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 57, and on the secondary server as described in “STEP SWCF4: Start Q Apply on secondary” on page 59.

STEP SWUFOPD5: Verify Q replication up and running

Verify Q replication is up and running on both servers as described in “STEP SWUFOPA17: Verify Q replication up and running” on page 80.

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 steps “STEP SWUFOPD6Ia: Verify Q Capture latency and E2E latency” on page 113 through “STEP SWUFOPD6Ig: Start Q Capture on secondary” on page 114.
- ▶ For loadphase ‘N’ execute “STEP SWUFOPD6Na: Resume primary workload” on page 114.

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 have been replicated over to the primary server as described in “STEP SWCF5: Verify Q Capture latency and E2E latency” on page 59.

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 redbook.

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 65.

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 65.

STEP SWUFOPD6ld: 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 66.

STEP SWUFOPD6le: 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 redbook.

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

STEP SWUFOPD6lf: Resume RO workload on secondary

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

STEP SWUFOPD6lg: Start Q Capture on secondary

Start Q Capture on the secondary server as described in “STEP SWCF3: Start Q Capture on secondary” on page 59.

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 redbook.

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 as shown in Example 3-47.

Example 3-47 Remove WebSphere MQ objects

```
endmqm -i QM_WEST  
endmqtsr -m QM_WEST  
dlrmqm -z QM_WEST
```

- Remove all remnants of the Q replication Capture and Apply control tables as by issuing the `asnc1p -f ASNCLP_drop_ctl.in` command as shown in Example 3-48.

Example 3-48 The ASNCLP_drop_ctl.in to remove the Q replication control tables on JAMAICA

```
# DROP CONTROL TABLES
ASNCLP SESSION SET TO Q REPLICATION;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
SET OUTPUT MULTIDIR;

SET SERVER TARGET TO DB WEST ID db2inst9 PASSWORD "xxx";

SET QMANAGER "QM_WEST" FOR APPLY SCHEMA;

SET APPLY SCHEMA WEST;

DROP CONTROL TABLES on APPLY SERVER;

SET SERVER CAPTURE TO DB WEST ID db2inst9 PASSWORD "xxx";

SET QMANAGER "QM_WEST" FOR CAPTURE SCHEMA;

SET CAPTURE SCHEMA SOURCE WEST;

DROP CONTROL TABLES on CAPTURE SERVER;
```

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 as shown in Example 3-49.

Note: Rather than delete the log files, you may choose to move them to a different directory from the Capture/Apply path, if further analysis of these logs is anticipated in future.

Example 3-49 Remove Q Capture and Q Apply logs

```
#Capture and Apply log naming rule
#Capture log => instance_name.DB_name.schema_name.QCAP.log
#Apply log   => instance_name.DB_name.schema_name.QAPP.log

#Remove Capture log
```

```
CAPTURE_PATH=/DB2  
cd ${CAPTURE_PATH}  
rm db2inst9.WEST.WEST.QCAP.log
```

```
#Remove Apply log  
APPLY_PATH=/DB2  
cd ${APPLY_PATH}  
rm db2inst9.WEST.WEST.QAPP.log
```

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.

HADR and Q replication coexistence scenario

In this chapter, we describe a high availability environment using HADR, into which is introduced a unidirectional replication scenario involving a data warehouse application.

The topics covered include:

- ▶ Business requirement
- ▶ Environment configuration
- ▶ Set up of the environment
- ▶ Planned takeover
- ▶ Unplanned takeover
- ▶ Failback

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 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 HADR high availability business solution implementation, into which is introduced seamlessly a unidirectional Q replication application. This unidirectional Q replication application coexists synergistically with the HADR high availability solution. A brief overview of the business requirement, is followed by a description of the environment configuration. The set up of this HADR 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
- ▶ Planned takeover
- ▶ Unplanned takeover
- ▶ Re-establish pre-takeover environment

4.2 Business requirement

Our fictitious company, FashionHits, 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 UDB for LUW implementation using HADR. 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 system.

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' HADR implementation.

4.3 Rationale for the unidirectional solution

IBM's WebSphere Replication Server provides the functionality of high volume low latency unidirectional replication for DB2 UDB for LUW data sources. Additionally, WebSphere Replication Server works seamlessly within an environment where the primary server of a Q replication scenario participates in a high availability HADR implementation.

4.4 Environment configuration

Figure 4-1 on page 120 shows the configuration used in the FashionHits unidirectional replication topology.

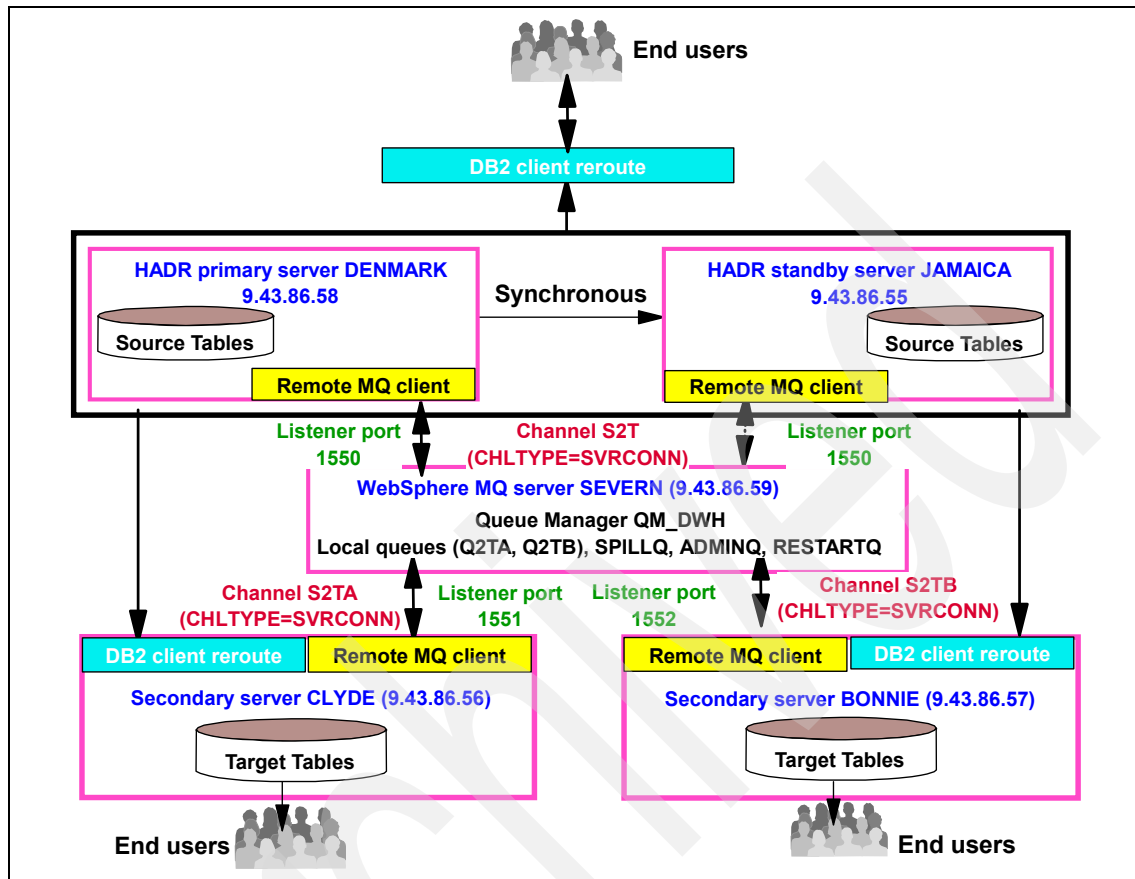


Figure 4-1 Unidirectional replication topology configuration

This topology may be broadly thought to consist of the following:

- ▶ High availability primary server environment using HADR and DB2 Client Reroute.
- ▶ Unidirectional replication between the high availability HADR primary server and two target servers

Each of these topology components are discussed here briefly:

4.4.1 HA primary environment using HADR and DB2 Client Reroute

In our FashionHits scenario, unidirectional replication is introduced into an existing high availability HADR environment with DB2 Client Reroute.

High Availability Disaster Recovery (HADR) is a DB2 UDB ESE feature that provides a high availability environment where the primary server and a standby server are kept in sync on an ongoing basis, so that in the event of the loss of the primary server, users can be switched manually or automatically to the standby server to provide almost uninterrupted service. HADR operates by shipping DB2 log records from the primary server to a standby server — with the standby server continually replaying the logs to keep the database synchronized with the primary server. A brief overview of HADR is provided in Appendix C.1, “HADR” on page 302.

The automatic DB2 Client Reroute feature is independent of the HADR capability, and allows client applications to recover from a loss of communication with the main server so that they can continue to work with minimal interruption. After a loss of communication with the main server, the client application attempts to reconnect to the it. If this fails, the client is then rerouted to an alternate server. A brief overview of DB2 Client Reroute is provided in Appendix C.2, “DB2 Client Reroute” on page 305.

These two features together provide significant high availability characteristics as shown in Figure 4-2.

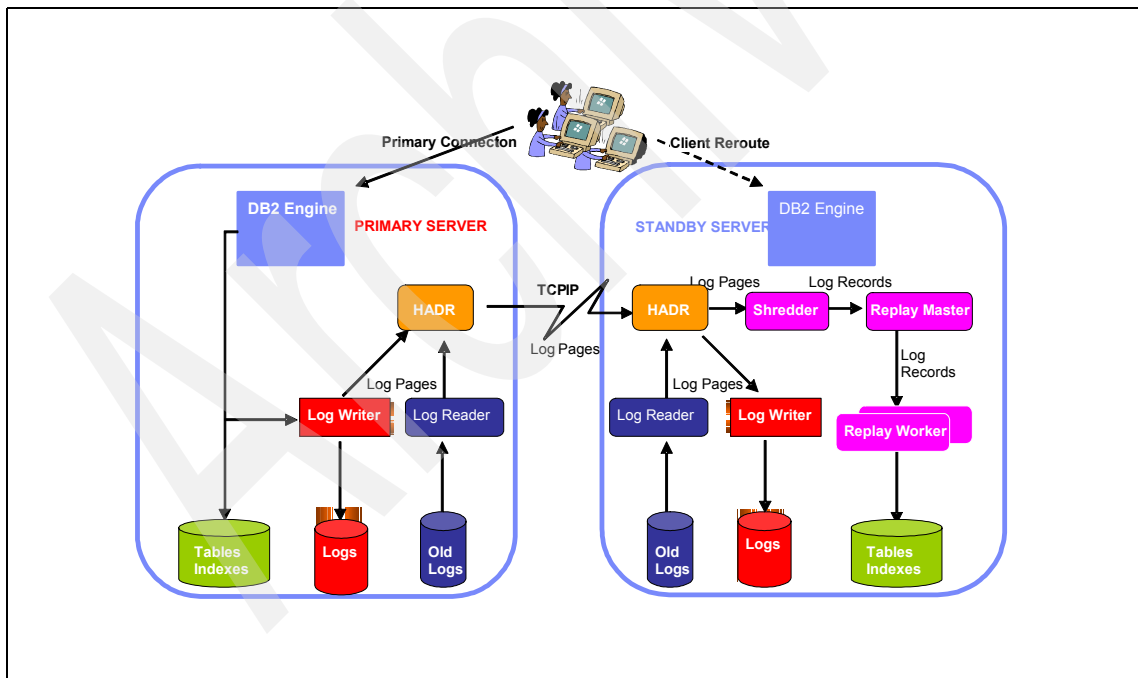


Figure 4-2 HADR and DB2 Client Reroute synergy

The typical flow is as follows:

1. Under normal operations, the users are connected to the primary server and execute transactions against the database on this server. HADR continuously captures the changes on the DB2 log, and ships it to the standby server where it is continuously replayed thereby keeping the target database in sync. The standby server database is in rollforward pending and no users access this standby server database.
2. After an outage on the primary server (either planned or unplanned), the database administrator issues a TAKEOVER HADR command against the standby server to change it from its standby status to the primary status — and this server can now serve users as the primary server.
3. While HADR takes care of synchronization of data between a primary/standby pair, there is still a need for applications to dynamically access the new primary server after an outage. This is achieved via the DB2 Client Reroute feature which requires the database administrator to specify each other as the alternate server on the primary server and the standby server.

When any client first connects to the primary server, the alternate server information is also sent to the client by DB2. Whenever the client fails to connect to the primary server, the client will attempt to reroute the connection to the alternate server. If the reroute succeeds, the role of the primary server and alternate server will reverse.

In the HADR scenario and DB2 Client Reroute scenario shown in Figure 4-2 on page 121, when an outage occurs of the primary server, and the standby server becomes the new primary server, DB2 Client Reroute ensures that users are automatically switched over to the new primary server.

4. When the failed server comes back on line, it can then be started as the standby server, reversing roles prior to the outage.

The following sections provide a brief overview of:

- ▶ Set up of HADR between the primary and standby server
- ▶ Set up DB2 Client Reroute on the primary and standby server

Set up HADR between the primary and standby server

In this section, we set up the HADR on a pair of servers DENMARK and JAMAICA, where DENMARK is the primary server and JAMAICA is the standby server.

Before setting up HADR, we implemented the following prerequisite steps:

1. Set up two cross mounted NFS file systems which DENMARK and JAMAICA share for DB2 backups and DB2 Logs as required of such a configuration. The file systems we used are /DB2/BACKUP and /DB2/LOGS
2. In our test environment, we ensured that only one backup existed for the database on the primary server by first removing all previous backups¹ and then performing a new DB2 backup — this avoided the need to specify the TAKEN AT backup image time in the RESTORE command. We took an ONLINE backup of the primary server database rather than an OFFLINE backup to minimize the window of unavailability of the application workload.
3. We used Port Number 50000 for all DB2 client connections on both servers (DENMARK and JAMAICA) by issuing the following command on each server:
db2 update dbm cfg using SVCENAME 50000

Figure 4-3 shows the steps required to set up HADR on DENMARK and JAMAICA.

Attention: The steps identified here represented the approach we took in our environment, but some of the steps may be performed in a different sequence.

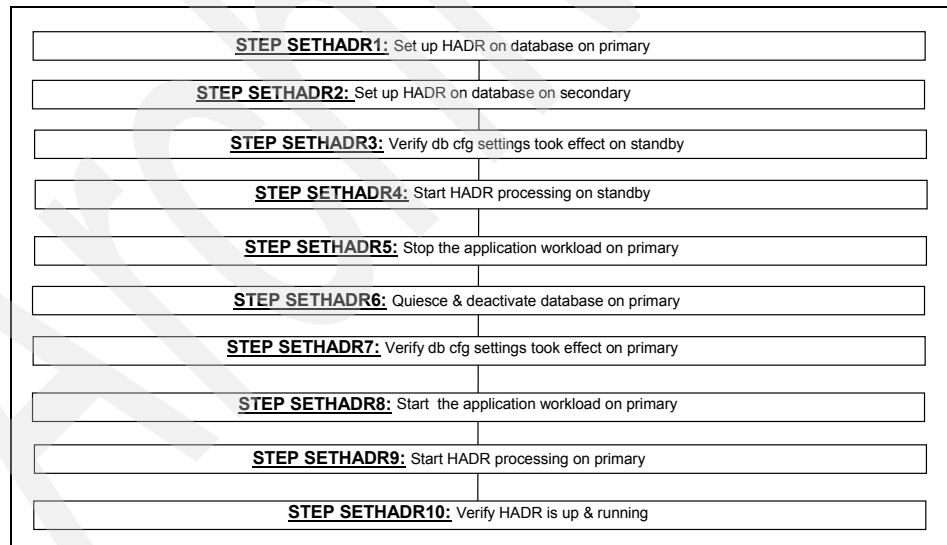


Figure 4-3 Overview of steps to set up HADR

¹ This would not be a realistic practice in a real world environment, but acceptable for our test environment.

The steps shown in Figure 4-3 on page 123 are described in the following sections.

STEP SETHADR1: Set up HADR on database on primary

In this step, run the script `setup_hadr_primary.sh` shown in Example 4-1 that deletes all existing backups, performs an online backup of the source database to the `/DB2/BACKUP` file system, and updates the source database configuration parameters on the primary server DENMARK. Synchronous mode is specified for the HADR environment.

Example 4-1 setup_hadr_primary.sh script on the primary server DENMARK

```
#!/bin/ksh

rm -fr /DB2/BACKUP/*

echo "
--
UPDATE DB CFG FOR SOURCE USING LOGFILSIZ 1000 ;
UPDATE DB CFG FOR SOURCE USING LOGPRIMARY 30 ;
UPDATE DB CFG FOR SOURCE USING INDEXREC RESTART ;
UPDATE DB CFG FOR SOURCE USING LOGARCHMETH1 DISK:/DB2/LOGS ;
--
BACKUP DB SOURCE ONLINE to "/DB2/BACKUP" WITHOUT PROMPTING;
--
CONNECT TO SOURCE;
UPDATE DB CFG FOR SOURCE USING HADR_SYNCMODE SYNC;
UPDATE DB CFG FOR SOURCE USING HADR_TIMEOUT 120;
UPDATE DB CFG FOR SOURCE USING HADR_LOCAL_HOST denmark;
UPDATE DB CFG FOR SOURCE USING HADR_LOCAL_SVC 60000;

UPDATE DB CFG FOR SOURCE USING HADR_REMOTE_HOST jamaica;
UPDATE DB CFG FOR SOURCE USING HADR_REMOTE_SVC 60001;
UPDATE DB CFG FOR SOURCE USING HADR_REMOTE_INST db2inst9;
--
CONNECT RESET;
" | db2 -tv
```

STEP SETHADR2: Set up HADR on database on secondary

In this step, run the script `setup_hadr_standby.sh` shown in Example 4-2 that restores the source database from the shared `/DB2/BACKUP` file system, and updates the source database configuration parameters on the secondary server JAMAICA. Synchronous mode is specified for the HADR environment. After the restore operation, the database is in rollforward pending state.

Example 4-2 setup_hadr_standby.sh script on the secondary server JAMAICA

```
#!/bin/ksh

echo "
RESTORE DB SOURCE FROM "/DB2/BACKUP"
REPLACE HISTORY FILE WITHOUT PROMPTING;
--
UPDATE DB CFG FOR SOURCE USING HADR_SYNCMODE SYNC;
UPDATE DB CFG FOR SOURCE USING HADR_TIMEOUT 120;
UPDATE DB CFG FOR SOURCE USING HADR_LOCAL_HOST jamaica;
UPDATE DB CFG FOR SOURCE USING HADR_LOCAL_SVC 60001;

UPDATE DB CFG FOR SOURCE USING HADR_REMOTE_HOST denmark;
UPDATE DB CFG FOR SOURCE USING HADR_REMOTE_SVC 60000;
UPDATE DB CFG FOR SOURCE USING HADR_REMOTE_INST db2inst9;
--
" | db2 -tv
```

STEP SETHADR3: Verify db cfg settings took effect on standby

In this step, verify that the configuration settings have taken effect by issuing the get db cfg for source command as shown in Example 4-3 and Example 4-4.

Note: Since the database is in rollforward pending state and no connections exist to the database, the configuration parameters take effect right away.

Example 4-3 Verify the source database cfg parameters on the standby server JAMAICA 1/2

db2 get db cfg for source;

Example 4-4 Verify the source database cfg parameters on the standby server JAMAICA 2/2

Database Configuration for Database source

| | |
|--|-------------|
| Database configuration release level | = 0x0b00 |
| Database release level | = 0x0b00 |
| Database territory | = US |
| Database code page | = 819 |
| Database code set | = ISO8859-1 |
| Database country/region code | = 1 |
| Database collating sequence | = UNIQUE |
| Alternate collating sequence (ALT_COLLATE) | = |
| Database page size | = 4096 |

| | |
|--|----------------------------------|
| Dynamic SQL Query management | (DYN_QUERY_MGMT) = DISABLE |
| Discovery support for this database | (DISCOVER_DB) = ENABLE |
| Restrict access | = NO |
| Default query optimization class | (DFT_QUERYOPT) = 5 |
| Degree of parallelism | (DFT_DEGREE) = 1 |
| Continue upon arithmetic exceptions | (DFT_SQLMATHWARN) = NO |
| Default refresh age | (DFT_REFRESH_AGE) = 0 |
| Default maintained table types for opt | (DFT_MTTB_TYPES) = SYSTEM |
| Number of frequent values retained | (NUM_FREQVALUES) = 10 |
| Number of quantiles retained | (NUM_QUANTILES) = 20 |
| Backup pending | = NO |
| Database is consistent | = YES |
| Rollforward pending | = DATABASE |
| Restore pending | = NO |
| Multi-page file allocation enabled | = YES |
| Log retain for recovery status | = NO |
| User exit for logging status | = YES |
| Self tuning memory | (SELF_TUNING_MEM) = ON |
| Size of database shared memory (4KB) | (DATABASE_MEMORY) = AUTOMATIC |
| Database memory threshold | (DB_MEM_THRESH) = 10 |
| Max storage for lock list (4KB) | (LOCKLIST) = 100 |
| Percent. of lock lists per application | (MAXLOCKS) = 10 |
| Package cache size (4KB) | (PCKCACHESZ) = (MAXAPPLS*8) |
| Sort heap thres for shared sorts (4KB) | (SHEAPTHRES_SHR) = 5000 |
| Sort list heap (4KB) | (SORTHEAP) = 256 |
| Database heap (4KB) | (DBHEAP) = 1200 |
| Catalog cache size (4KB) | (CATALOGCACHE_SZ) = (MAXAPPLS*4) |
| Log buffer size (4KB) | (LOGBUFSZ) = 8 |
| Utilities heap size (4KB) | (UTIL_HEAP_SZ) = 5000 |
| Buffer pool size (pages) | (BUFFPAGE) = 1000 |
| Max size of appl. group mem set (4KB) | (APPGROUP_MEM_SZ) = 30000 |
| Percent of mem for appl. group heap | (GROUPHEAP_RATIO) = 70 |
| Max appl. control heap size (4KB) | (APP_CTL_HEAP_SZ) = 128 |
| SQL statement heap (4KB) | (STMTHEAP) = 4096 |
| Default application heap (4KB) | (APPLHEAPSZ) = 256 |
| Statistics heap size (4KB) | (STAT_HEAP_SZ) = 4384 |
| Interval for checking deadlock (ms) | (DLCHKTIME) = 10000 |
| Lock timeout (sec) | (LOCKTIMEOUT) = -1 |

| | |
|--|--------------------------------------|
| Changed pages threshold | (CHNGPGS_THRESH) = 60 |
| Number of asynchronous page cleaners | (NUM_IOCLEANERS) = AUTOMATIC |
| Number of I/O servers | (NUM_IOSERVERS) = AUTOMATIC |
| Index sort flag | (INDEXSORT) = YES |
| Sequential detect flag | (SEQDETECT) = YES |
| Default prefetch size (pages) | (DFT_PREFETCH_SZ) = AUTOMATIC |
| | |
| Track modified pages | (TRACKMOD) = OFF |
| Default number of containers | = 1 |
| Default tablespace extentsize (pages) | (DFT_EXTENT_SZ) = 32 |
| | |
| Max number of active applications | (MAXAPPLS) = AUTOMATIC |
| Average number of active applications | (AVG_APPLS) = AUTOMATIC |
| Max DB files open per application | (MAXFILOP) = 64 |
| | |
| Log file size (4KB) | (LOGFILSIZ) = 1000 |
| Number of primary log files | (LOGPRIMARY) = 30 |
| Number of secondary log files | (LOGSECOND) = 2 |
| Changed path to log files | (NEWLOGPATH) = |
| Path to log files | = |
| /DB2/db2inst9/NODE0000/SQL00003/SQLLOGDIR/ | |
| Overflow log path | (OVERFLOWLOGPATH) = |
| Mirror log path | (MIRRORLOGPATH) = |
| First active log file | = S0000116.LOG |
| Block log on disk full | (BLK_LOG_DSK_FUL) = NO |
| Percent of max active log space by transaction | (MAX_LOG) = 0 |
| Num. of active log files for 1 active UOW | (NUM_LOG_SPAN) = 0 |
| | |
| Group commit count | (MINCOMMIT) = 1 |
| Percent log file reclaimed before soft ckcpt | (SOFTMAX) = 100 |
| Log retain for recovery enabled | (LOGRETAIN) = OFF |
| User exit for logging enabled | (USEREXIT) = OFF |
| | |
| HADR database role | = STANDARD |
| HADR local host name | (HADR_LOCAL_HOST) = jamaica |
| HADR local service name | (HADR_LOCAL_SVC) = 60001 |
| HADR remote host name | (HADR_REMOTE_HOST) = denmark |
| HADR remote service name | (HADR_REMOTE_SVC) = 60000 |
| HADR instance name of remote server | (HADR_REMOTE_INST) = db2inst9 |
| HADR timeout value | (HADR_TIMEOUT) = 120 |
| HADR log write synchronization mode | (HADR_SYNCMODE) = SYNC |
| | |
| First log archive method | (LOGARCHMETH1) = DISK:/DB2/LOGS/ |
| Options for logarchmeth1 | (LOGARCHOPT1) = |
| Second log archive method | (LOGARCHMETH2) = OFF |
| Options for logarchmeth2 | (LOGARCHOPT2) = |
| Failover log archive path | (FAILARCHPATH) = |

| | |
|---|-------------------------|
| Number of log archive retries on error | (NUMARCHRETRY) = 5 |
| Log archive retry Delay (secs) | (ARCHRETRYDELAY) = 20 |
| Vendor options | (VENDOROPT) = |
| Auto restart enabled | (AUTORESTART) = ON |
| Index re-creation time and redo index build | (INDEXREC) = RESTART |
| Log pages during index build | (LOGINDEXBUILD) = ON |
| Default number of loadrec sessions | (DFT_LOADREC_SES) = 1 |
| Number of database backups to retain | (NUM_DB_BACKUPS) = 12 |
| Recovery history retention (days) | (REC_HIS_RETENTN) = 366 |
| TSM management class | (TSM_MGMTCLASS) = |
| TSM node name | (TSM_NODENAME) = |
| TSM owner | (TSM_OWNER) = |
| TSM password | (TSM_PASSWORD) = |
| Automatic maintenance | (AUTO_MAINT) = ON |
| Automatic database backup | (AUTO_DB_BACKUP) = OFF |
| Automatic table maintenance | (AUTO_TBL_MAINT) = ON |
| Automatic runstats | (AUTO_RUNSTATS) = ON |
| Automatic statistics profiling | (AUTO_STATS_PROF) = OFF |
| Automatic profile updates | (AUTO_PROF_UPD) = OFF |
| Automatic reorganization | (AUTO_REORG) = OFF |

STEP SETHADR4: Start HADR processing on standby

In this step, run the script `start_hadr_standby.sh` shown in Example 4-5 to start HADR processing on the server JAMAICA as a standby.

Attention: We recommend that you start the database on the standby server before the corresponding one on the primary server. If you start the primary database first, the startup procedure will fail if the standby database is not started within the time period specified by the `HADR_TIMEOUT` database configuration parameter.

Example 4-5 start_hadr_standby.sh script on the secondary server JAMAICA

```
#!/bin/ksh

echo "
START HADR ON DATABASE SOURCE AS STANDBY;
" | db2 -tv
```

STEP SETHADR5: Stop application workload on primary

In this step, stop the application workload on the primary server DENMARK. The process used to achieve this varies from organization to organization and is beyond the scope of this redbook.

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

STEP SETHADR6: Quiesce and deactivate database on primary

In this step, run the `deactivate_primary.sh` script shown in Example 4-6 that quiesces (by force to terminate any existing connections), deactivates and then unquiesces the source database to ensure that the update of the HADR configuration parameters have taken effect.

Note: This is required because the updates to the database configuration parameters cannot take effect until all the connections from the ongoing application workload on the primary server are terminated.

Example 4-6 deactivate_primary.sh script on the primary server DENMARK

```
#!/bin/ksh

echo "
--
-- make sure your workload has been stopped before, otherwise it will be forced
--
CONNECT TO SOURCE;
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
CONNECT RESET;
--
DEACTIVATE DATABASE SOURCE;
--
--
-- UNQUIESCE the database ...
--
CONNECT TO SOURCE;
UNQUIESCE DATABASE;
CONNECT RESET;
--
" | db2 -tv
```

STEP SETHADR7: Verify db cfg settings took effect on primary

In this step, run the SQL shown in Example 4-7 that verifies the database configuration settings on the source database have taken effect as shown in Example 4-8 on page 130.

Example 4-7 Verify db cfg setting on the primary server DENMARK

```
db2 connect to source;  
db2 get db cfg for source show detail;  
db2 terminate;
```

Example 4-8 Verify db cfg setting on the primary server DENMARK

db2 connect to source;

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = SOURCE

db2 get db cfg for source show detail;

Database Configuration for Database source

| Description | Parameter | Current Value | Delayed Value |
|--|-------------------|---------------|---------------|
| Database configuration release level | | = 0x0b00 | |
| Database release level | | = 0x0b00 | |
| Database territory | | = US | |
| Database code page | | = 819 | |
| Database code set | | = ISO8859-1 | |
| Database country/region code | | = 1 | |
| Database collating sequence | | = UNIQUE | UNIQUE |
| Alternate collating sequence | (ALT_COLLATE) | = | |
| Database page size | | = 4096 | 4096 |
| Dynamic SQL Query management | (DYN_QUERY_MGMT) | = DISABLE | DISABLE |
| Discovery support for this database | (DISCOVER_DB) | = ENABLE | ENABLE |
| Restrict access | | = NO | |
| Default query optimization class | (DFT_QUERYOPT) | = 5 | 5 |
| Degree of parallelism | (DFT_DEGREE) | = 1 | 1 |
| Continue upon arithmetic exceptions | (DFT_SQLMATHWARN) | = NO | NO |
| Default refresh age | (DFT_REFRESH_AGE) | = 0 | 0 |
| Default maintained table types for opt | (DFT_MTTB_TYPES) | = SYSTEM | SYSTEM |
| Number of frequent values retained | (NUM_FREQVALUES) | = 10 | 10 |
| Number of quantiles retained | (NUM_QUANTILES) | = 20 | 20 |
| Backup pending | | = NO | |
| Database is consistent | | = YES | |
| Rollforward pending | | = NO | |
| Restore pending | | = NO | |

| | | |
|--|--|------------------|
| Multi-page file allocation enabled | = YES | |
| Log retain for recovery status | = NO | |
| User exit for logging status | = YES | |
| Self tuning memory | (SELF_TUNING_MEM) = ON (Inactive) | ON |
| Size of database shared memory (4KB) | (DATABASE_MEMORY) = AUTOMATIC(18208) | AUTOMATIC(18208) |
| Database memory threshold | (DB_MEM_THRESH) = 10 | 10 |
| Max storage for lock list (4KB) | (LOCKLIST) = 100 | 100 |
| Percent. of lock lists per application | (MAXLOCKS) = 10 | 10 |
| Package cache size (4KB) | (PCKCACHESZ) = (MAXAPPLS*8) | (MAXAPPLS*8) |
| Sort heap thres for shared sorts (4KB) | (SHEAPTHRES_SHR) = 5000 | 5000 |
| Sort list heap (4KB) | (SORTHEAP) = 256 | 256 |
| Database heap (4KB) | (DBHEAP) = 1200 | 1200 |
| Catalog cache size (4KB) | (CATALOGCACHE_SZ) = (MAXAPPLS*4) | (MAXAPPLS*4) |
| Log buffer size (4KB) | (LOGBUF SZ) = 8 | 8 |
| Utilities heap size (4KB) | (UTIL_HEAP_SZ) = 5000 | 5000 |
| Buffer pool size (pages) | (BUFFPAGE) = 1000 | 1000 |
| Max size of appl. group mem set (4KB) | (APPGROUP_MEM_SZ) = 30000 | 30000 |
| Percent of mem for appl. group heap | (GROUPHEAP_RATIO) = 70 | 70 |
| Max appl. control heap size (4KB) | (APP_CTL_HEAP_SZ) = 128 | 128 |
| SQL statement heap (4KB) | (STMTHEAP) = 4096 | 4096 |
| Default application heap (4KB) | (APPLHEAPSZ) = 256 | 256 |
| Statistics heap size (4KB) | (STAT_HEAP_SZ) = 4384 | 4384 |
| Interval for checking deadlock (ms) | (DLCHKTIME) = 10000 | 10000 |
| Lock timeout (sec) | (LOCKTIMEOUT) = -1 | -1 |
| Changed pages threshold | (CHNGPGS_THRESH) = 60 | 60 |
| Number of asynchronous page cleaners | (NUM_IOCLEANERS) = AUTOMATIC(3) | AUTOMATIC(3) |
| Number of I/O servers | (NUM_IOSERVERS) = AUTOMATIC(3) | AUTOMATIC(3) |
| Index sort flag | (INDEXSORT) = YES | YES |
| Sequential detect flag | (SEQDETECT) = YES | YES |
| Default prefetch size (pages) | (DFT_PREFETCH_SZ) = AUTOMATIC | AUTOMATIC |
| Track modified pages | (TRACKMOD) = NO | NO |
| Default number of containers | = 1 | 1 |
| Default tablespace extentsize (pages) | (DFT_EXTENT_SZ) = 32 | 32 |
| Max number of active applications | (MAXAPPLS) = AUTOMATIC(40) | AUTOMATIC(40) |
| Average number of active applications | (AVG_APPLS) = AUTOMATIC(1) | AUTOMATIC(1) |
| Max DB files open per application | (MAXFILOP) = 64 | 64 |
| Log file size (4KB) | (LOGFILSIZ) = 1000 | 1000 |
| Number of primary log files | (LOGPRIMARY) = 30 | 30 |
| Number of secondary log files | (LOGSECOND) = 2 | 2 |
| Changed path to log files | (NEWLOGPATH) = | |
| Path to log files | = /DB2/db2inst9/NODE0000/SQL00004/SQLLOGDIR/ | |
| /DB2/db2inst9/NODE0000/SQL00004/SQLLOGDIR/ | | |
| Overflow log path | (OVERFLOWLOGPATH) = | |
| Mirror log path | (MIRRORLOGPATH) = | |

| | | |
|--|----------------------------------|-----------------|
| First active log file | = S0000117.LOG | S0000117.LOG |
| Block log on disk full | (BLK_LOG_DSK_FUL) = NO | NO |
| Percent of max active log space by transaction | (MAX_LOG) = 0 | 0 |
| Num. of active log files for 1 active UOW | (NUM_LOG_SPAN) = 0 | 0 |
| Group commit count | (MINCOMMIT) = 1 | 1 |
| Percent log file reclaimed before soft ckckpt | (SOFTMAX) = 100 | 100 |
| Log retain for recovery enabled | (LOGRETAIN) = OFF | OFF |
| User exit for logging enabled | (USEREXIT) = OFF | OFF |
| | | |
| HADR database role | = STANDARD | STANDARD |
| HADR local host name | (HADR_LOCAL_HOST) = denmark | denmark |
| HADR local service name | (HADR_LOCAL_SVC) = 60000 | 60000 |
| HADR remote host name | (HADR_REMOTE_HOST) = jamaica | jamaica |
| HADR remote service name | (HADR_REMOTE_SVC) = 60001 | 60001 |
| HADR instance name of remote server | (HADR_REMOTE_INST) = db2inst9 | db2inst9 |
| HADR timeout value | (HADR_TIMEOUT) = 120 | 120 |
| HADR log write synchronization mode | (HADR_SYNCMODE) = SYNC | SYNC |
| | | |
| First log archive method | (LOGARCHMETH1) = DISK:/DB2/LOGS/ | DISK:/DB2/LOGS/ |
| Options for logarchmeth1 | (LOGARCHOPT1) = | |
| Second log archive method | (LOGARCHMETH2) = OFF | OFF |
| Options for logarchmeth2 | (LOGARCHOPT2) = | |
| Failover log archive path | (FAILARCHPATH) = | |
| Number of log archive retries on error | (NUMARCHRETRY) = 5 | 5 |
| Log archive retry Delay (secs) | (ARCHRETRYDELAY) = 20 | 20 |
| Vendor options | (VENDOROPT) = | |
| Auto restart enabled | (AUTORESTART) = ON | ON |
| Index re-creation time and redo index build | (INDEXREC) = RESTART | RESTART |
| Log pages during index build | (LOGINDEXBUILD) = ON | ON |
| Default number of loadrec sessions | (DFT_LOADREC_SES) = 1 | 1 |
| Number of database backups to retain | (NUM_DB_BACKUPS) = 12 | 12 |
| Recovery history retention (days) | (REC_HIS_RETENTN) = 366 | 366 |
| | | |
| TSM management class | (TSM_MGMTCLASS) = | |
| TSM node name | (TSM_NODENAME) = | |
| TSM owner | (TSM_OWNER) = | |
| TSM password | (TSM_PASSWORD) = | |
| | | |
| Automatic maintenance | (AUTO_MAINT) = ON | ON |
| Automatic database backup | (AUTO_DB_BACKUP) = OFF | OFF |
| Automatic table maintenance | (AUTO_TBL_MAINT) = ON | ON |
| Automatic runstats | (AUTO_RUNSTATS) = ON | ON |
| Automatic statistics profiling | (AUTO_STATS_PROF) = OFF | OFF |
| Automatic profile updates | (AUTO_PROF_UPD) = OFF | OFF |
| Automatic reorganization | (AUTO_REORG) = OFF | OFF |

db2 terminate;

DB20000I The TERMINATE command completed successfully.

STEP SETHADR8: Start application workload on primary

In this step, resume the application workload on the primary server DENMARK. The process used to achieve this varies from organization to organization and is beyond the scope of this redbook.

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

STEP SETHADR9: Start HADR processing on primary

In this step, run the start_hadr_primary.sh script as shown in Example 4-9 that starts HADR on the source database as the primary server.

Example 4-9 start_hadr_primary script on the primary server DENMARK

```
#!/bin/ksh

echo "
--
START HADR ON DATABASE SOURCE AS PRIMARY;
--
" | db2 -tv
```

STEP SETHADR10: Verify HADR is up and running

In this step, issue the db2pd command that verifies that the primary and standby database are catching up — HADR state becomes ‘Peer’ on both servers as shown in Example 4-10 and Example 4-11 on page 134.

Example 4-10 Check HADR status on the primary server DENMARK

```
db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 0 days 02:38:49

HADR Information:
Role      State      SyncMode HeartBeatsMissed LogGapRunAvg (bytes)
Primary Peer      Sync      0                0

ConnectStatus ConnectTime Timeout
Connected Tue Mar 7 10:22:36 2006 (1141748556) 120

LocalHost LocalService
denmark 60000

RemoteHost RemoteService RemoteInstance
jamaica 60001 db2inst9

PrimaryFile PrimaryPg PrimaryLSN
```

```
S0000071.LOG 495      0x00000000122FFF64

StandByFile StandByPg StandByLSN
S0000071.LOG 495      0x00000000122FFF64
```

Example 4-11 Check HADR status on the standby server JAMAICA

```
db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 3 days 21:07:05

HADR Information:
Role      State      SyncMode HeartBeatsMissed LogGapRunAvg (bytes)
Standby Peer      Sync      0              0

ConnectStatus ConnectTime      Timeout
Connected      Tue Mar 7 10:22:36 2006 (1141748556) 120

LocalHost      LocalService
jamaica        60001

RemoteHost      RemoteService RemoteInstance
denmark         60000         db2inst9

PrimaryFile PrimaryPg PrimaryLSN
S0000071.LOG 495      0x00000000122FFF64

StandByFile StandByPg StandByLSN
S0000071.LOG 495      0x00000000122FFF64
```

Set up DB2 Client Reroute

In this section, we set up DB2 Client Reroute on servers DENMARK and JAMAICA, where DENMARK is the primary server and JAMAICA is the standby server.

Figure 4-4 shows the steps required to set up DB2 Client Reroute in our scenario.

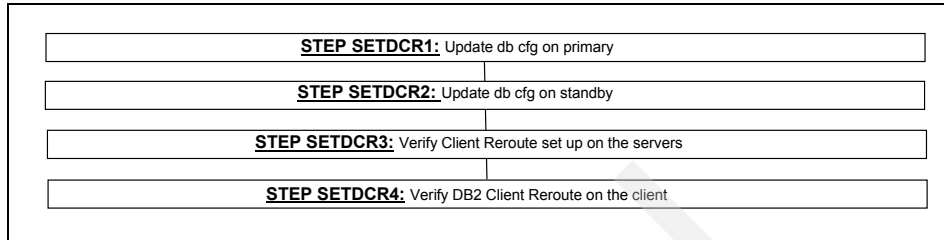


Figure 4-4 Overview of steps to set up DB2 Client Reroute

The steps shown in Figure 4-4 are described in the following sections.

STEP SETDCR1: Update db cfg on primary

In this step, run the `update_alternate_server_primary.sh` script as shown in Example 4-12 that updates the source database configuration settings on the primary server DENMARK to identify the standby server JAMAICA as the alternate server.

Example 4-12 The `update_alternate_server_primary.sh` script on the primary server DENMARK

```

echo "
UPDATE ALTERNATE SERVER FOR DATABASE SOURCE
USING HOSTNAME jamaica PORT 50000;
TERMINATE;
" | db2 -tv
  
```

STEP SETDCR2: Update db cfg on standby

In this step, run the `update_alternate_server_standby.sh` script as shown in Example 4-13 that updates the source database configuration settings on the standby server JAMAICA to identify the primary server DENMARK as the alternate server.

Example 4-13 The `update_alternate_server_primary.sh` script on the standby server JAMAICA

```

echo "
UPDATE ALTERNATE SERVER FOR DATABASE SOURCE
USING HOSTNAME denmark PORT 50000;
TERMINATE;
" | db2 -tv
  
```

STEP SETDCR3: Verify Client Reroute set up on the servers

In this step, verify the DB2 Client Reroute set up on the primary and secondary server on each server using the `list db directory` command.

Example 4-14 shows the alternate server hostname of JAMAICA on the primary server DENMARK.

Example 4-15 on page 136 shows the alternate server hostname of DENMARK on the standby server JAMAICA.

Example 4-14 LIST DB DIRECTORY on the primary server DENMARK

DB2 LIST DB DIRECTORY

System Database Directory

Number of entries in the directory = 7

Database 1 entry:

| | |
|-------------------------------------|------------------|
| Database alias | = SOURCE |
| Database name | = SOURCE |
| Local database directory | = /DB2 |
| Database release level | = b.00 |
| Comment | = |
| Directory entry type | = Indirect |
| Catalog database partition number | = 0 |
| Alternate server hostname | = jamaica |
| Alternate server port number | = 50000 |

[...]

Example 4-15 LIST DB DIRECTORY on the standby server JAMAICA

DB2 LIST DB DIRECTORY

System Database Directory

Number of entries in the directory = 5

Database 1 entry:

| | |
|-------------------------------------|------------------|
| Database alias | = SOURCE |
| Database name | = SOURCE |
| Local database directory | = /DB2 |
| Database release level | = b.00 |
| Comment | = |
| Directory entry type | = Indirect |
| Catalog database partition number | = 0 |
| Alternate server hostname | = denmark |
| Alternate server port number | = 50000 |

[...]

STEP SETDCR4: Verify DB2 Client Reroute on the client

In this step, verify that the DB2 Client Reroute works from a client. The following steps demonstrate the successful operation of DB2 Client Reroute.

1. Catalog the remote TCP IP node on the client as shown in Example 4-16.

Example 4-16 Catalog primary server DENMARK node on the client

db2 CATALOG TCPIP NODE denmark REMOTE denmark SERVER 50000

DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.

2. Catalog the remote database SOURCE on the primary server DENMARK on the client as shown in Example 4-17.

Example 4-17 Catalog primary server DENMARK database SOURCE on the client

db2 CATALOG DATABASE SOURCE AT NODE DENMARK

DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.

3. List the contents of the database directory on the client as shown in Example 4-18.

It shows the Alternate server hostname as being blank since no connection has as yet been established with the database on the primary server DENMARK.

Example 4-18 LIST DB DIRECTORY on the client

db2 LIST DB DIRECTORY

System Database Directory

Number of entries in the directory = 4

Database 1 entry:

| | |
|------------------------|-----------|
| Database alias | = SOURCE |
| Database name | = SOURCE |
| Node name | = DENMARK |
| Database release level | = a.00 |
| Comment | = |

| | |
|-------------------------------------|----------|
| Directory entry type | = Remote |
| Catalog database partition number | = -1 |
| Alternate server hostname | = |
| Alternate server port number | = |

.....

4. Connect to the SOURCE database from the client as shown in Example 4-19.

Example 4-19 Connect to SOURCE database from the client

```
db2 CONNECT TO source USER db2inst9 USING xxx
```

| | |
|---------------------------------|-------------------|
| Database Connection Information | |
| Database server | = DB2/AIX64 9.1.0 |
| SQL authorization ID | = DB2INST9 |
| Local database alias | = SOURCE |

5. Again list the contents of the database directory on the client as shown in Example 4-20.

This time it shows the Alternate server hostname as being JAMAICA since the connection established in the previous step causes this information to be transferred from the primary server DENMARK to the client.

Example 4-20 LIST DB DIRECTORY on the client

```
db2 LIST DB DIRECTORY
```

System Database Directory

Number of entries in the directory = 4

| | |
|-------------------------------------|------------------|
| Database 1 entry: | |
| Database alias | = SOURCE |
| Database name | = SOURCE |
| Node name | = DENMARK |
| Database release level | = a.00 |
| Comment | = |
| Directory entry type | = Remote |
| Catalog database partition number | = -1 |
| Alternate server hostname | = jamaica |
| Alternate server port number | = 50000 |

.....

6. Issue get connection state to show the connection state of the client as shown in Example 4-21.

It indicates that the client is connected to database SOURCE on Hostname DENMARK.

Example 4-21 get connection state

db2 GET CONNECTION STATE

Database Connection State
Connection state = **Connectable and Connected**
Connection mode = SHARE
Local database alias = SOURCE
Database name = **SOURCE**
Hostname = **denmark**
Service name = 50000

7. Issue db2 list applications on the primary server DENMARK to confirm that the client G92B7027 is connected to database SOURCE on this server as shown in Example 4-22.

Example 4-22 db2 list applications on the primary server DENMARK

db2 list applications

| Auth Id | Application Name | Appl. Handle | Application Id | DB Name | # of Agents |
|----------|------------------|--------------|----------------------------|---------|-------------|
| DB2INST9 | db2bp.exe | 116 | G92B7027.L10A.00C4C9160239 | SOURCE | 1 |

8. View the contents of the PRODUCTS table as shown in Example 4-23.

Example 4-23 Select statement issued from the client

db2 "SELECT * FROM PRODUCTS"

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|-------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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.

9. Perform a controlled failover on the HADR pair by issuing the TAKEOVER command on the standby server JAMAICA as shown in Example 4-24.
- After this command has taken effect, the standby server JAMAICA becomes the new primary server.

db2 TAKEOVER HADR ON DATABASE SOURCE

10. Once again issue a SELECT statement to view the contents of the PRODUCTS table as shown in Example 4-25 on page 140.

This initial attempt to connect to the SOURCE database on the primary server DENMARK fails, and DB2 attempts to connect to the alternate server JAMAICA and succeeds. It then returns the SQL30108N message.

The db2diag.log contents shown in Example 4-26 on page 140 document the progress of Client Reroute.

Example 4-25 Select statement issued from the client

db2 "SELECT * FROM PRODUCTS"

SQL30108N A connection failed but has been re-established. The hostname or IP address is "jamaica" and the service name or port number is "50000". Special registers may or may not be re-attempted (Reason code = "1"). SQLSTATE=08506

Example 4-26 db2diag.log contents on the client

2006-03-09-08.06.26.491000-480 I407242H1019 LEVEL: Warning
PID : 2396 TID : 752 PROC : db2bp.exe
INSTANCE: DB2 NODE : 000
APPID : G92B7027.L10A.00C4C9160239
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrRetrySetup, probe:10
MESSAGE : Client Reroute is starting....

DATA #1 : Hexdump, 136 bytes

| | | |
|------------|---|------------------|
| 0x0012FB30 | : 5351 4C43 4120 2020 8800 0000 7F8A FFFF | SQLCA |
| 0x0012FB40 | : 2500 2AFF 2AFF 30FF 5443 502F 4950 FF53 | %.*.*.0.TCP/IP.S |
| 0x0012FB50 | : 4F43 4B45 5453 FF39 2E34 332E 3836 2E35 | OCKETS.9.43.86.5 |
| 0x0012FB60 | : 38FF 7265 6376 FF20 2020 2020 2020 2020 | 8.recv. |
| 0x0012FB70 | : 2020 2020 2020 2020 2020 2020 2020 2020 | |
| 0x0012FB80 | : 2020 2020 2020 2020 5351 4C4A 434D 4E20 | SQLJCMN |
| 0x0012FB90 | : 1200 3681 1200 0000 0000 0000 0000 0000 | ..6..... |
| 0x0012FBA0 | : 0000 0000 0000 0000 2020 2020 2020 2020 | |
| 0x0012FBB0 | : 2020 2020 2020 2020 | |

2006-03-09-08.06.26.501000-480 I408263H1558 LEVEL: Warning
PID : 2396 TID : 752 PROC : db2bp.exe
INSTANCE: DB2 NODE : 000
APPID : G92B7027.L10A.00C4C9160239
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20
MESSAGE : Reconnecting to Hostname/IP Address -->
DATA #1 : Hexdump, 256 bytes

```

0x01584C7E : 4465 6E6D 6172 6B2E 6974 736F 736A 2E73   Denmark.itsosj.s
0x01584C8E : 616E 6A6F 7365 2E69 626D 2E63 6F6D 0000   anjose.ibm.com..
0x01584C9E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CAE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CBE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CCE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CDE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CEE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CFE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D0E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D1E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D2E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D3E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D4E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D5E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D6E : 0000 0000 0000 0000 0000 0000 0000 0000   .....

```

```

2006-03-09-08.06.26.511000-480 I409823H451          LEVEL: Warning
PID      : 2396                      TID   : 752        PROC  : db2bp.exe
INSTANCE: DB2                      NODE  : 000
APPID    : G92B7027.L10A.00C4C9160239
FUNCTION: DB2 UDB, DRDA Application Requester, sqljlrListToConnect, probe:20
MESSAGE  : Reconnecting to Service name/Port number -->
DATA #1 : Hexdump, 15 bytes
0x01584D7E : 3530 3030 3000 0000 0000 0000 0000 00    50000.....

```

```

2006-03-09-08.06.26.551000-480 I410276H1558          LEVEL: Warning
PID      : 2396                      TID   : 752        PROC  : db2bp.exe
INSTANCE: DB2                      NODE  : 000
APPID    : G92B7027.L70A.00C4C9160626
FUNCTION: DB2 UDB, DRDA Application Requester, sqljlrListToConnect, probe:20
MESSAGE  : Reconnecting to Hostname/IP Address -->
DATA #1 : Hexdump, 256 bytes
0x01584C7E : 6A61 6D61 6963 6100 0000 0000 0000 0000   jamaica.....
0x01584C8E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584C9E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CAE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CBE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CCE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CDE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CEE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CFE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D0E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D1E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D2E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D3E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D4E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D5E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D6E : 0000 0000 0000 0000 0000 0000 0000 0000   .....

```

```

2006-03-09-08.06.26.561000-480 I411836H451      LEVEL: Warning
PID      : 2396                      TID   : 752      PROC  : db2bp.exe
INSTANCE: DB2                      NODE   : 000
APPID    : G92B7027.L70A.00C4C9160626
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20
MESSAGE  : Reconnecting to Service name/Port number -->
DATA #1 : Hexdump, 15 bytes
0x01584D7E : 3530 3030 3000 0000 0000 0000 0000 00      50000.....

```

```

2006-03-09-08.06.26.631000-480 I412289H346      LEVEL: Warning
PID      : 2396                      TID   : 752      PROC  : db2bp.exe
INSTANCE: DB2                      NODE   : 000
APPID    : G92B7027.L80A.00C4C9160628
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrClientReroute, probe:998
MESSAGE  : Client Reroute is completed successfully.

```

11. Again list the contents of the database directory on the client as shown in Example 4-27.

This time it shows the Alternate server hostname as being DENMARK since the role of primary has flipped after the takeover and the successful Client Reroute connection.

Example 4-27 LIST DB DIRECTORY on the client

db2 LIST DB DIRECTORY

System Database Directory

Number of entries in the directory = 4

Database 1 entry:

```

Database alias      = SOURCE
Database name      = SOURCE
Node name          = DENMARK
Database release level = a.00
Comment            =
Directory entry type = Remote
Catalog database partition number = -1
Alternate server hostname = denmark
Alternate server port number = 50000

```

.....

12. Issue get connection state to view the connection state of the client as shown in Example 4-28.

It now indicates that the client is connected to database SOURCE on Hostname JAMAICA.

Example 4-28 Get Connection State on the client

db2 GET CONNECTION STATE

Database Connection State
Connection state = Connectable and Connected
Connection mode = SHARE
Local database alias = SOURCE
Database name = SOURCE
Hostname = **jamaica**
Service name = **50000**

13. Issue db2 list applications on the primary server JAMAICA to confirm that the client G92B7027 is connected to database SOURCE on this server as shown in Example 4-29.

Example 4-29 db2 list applications on the standby server JAMAICA

db2 list applications

| Auth Id | Application Name | Appl. Handle | Application Id | DB Name | # of Agents |
|----------|------------------|--------------|------------------------------------|---------|-------------|
| DB2INST9 | db2bp.exe | 568 | G92B7027 .L80A.00C4C9160628 | SOURCE | 1 |

14. View the contents of the PRODUCTS table as shown in Example 4-30 to verify that DB2 Client Reroute occurred transparently and successfully.

Example 4-30 Select statement issued from the client

db2 "SELECT * FROM PRODUCTS"

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|-------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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.

4.4.2 Unidirectional replication environment

The primary server DENMARK (IP address 9.43.86.58) has a standby server JAMAICA (IP address 9.43.86.55) that is kept in sync using HADR synchronous mode. The primary server database is SOURCE. The Q Capture process is started on the primary server DENMARK.

The WebSphere MQ server is installed on a separate server SEVERN (IP address 9.43.86.59). The Q replication queue objects such as the local queues (Q2TA and Q2TB), spill queue (SPILLQ), restart queue (RESTARTQ), admin queue (ADMINQ) and channels (S2T, S2TA and S2TB) are defined on the WebSphere MQ server SEVERN. The various listener ports used are identified in Figure 4-1 on page 120.

Important: In this scenario, we set up WebSphere MQ on a separate server — there is a performance penalty in doing so since the WebSphere MQ clients (Q Capture and Q Apply) on the primary server, standby server (after takeover), and secondary servers have to go across a network to access the queues. This WebSphere MQ server isolation also introduces it as a single point of failure — this can be addressed by implementing HA for WebSphere MQ, a discussion of which is beyond the scope of this redbook.

One way of reducing the performance penalty is to collocate WebSphere MQ on one of the secondary servers, to reduce the cost of network access to the queues from at least that secondary server.

Two secondary servers are implemented as follows:

- ▶ The secondary server CLYDE (IP address 9.43.86.56) has the database TARGETA. A Q Apply process is started on this server.
- ▶ The secondary server BONNIE (IP address 9.43.86.57) has the database TARGETB. A Q Apply process is started on this server.

We installed a set of two tables on the primary server, and replicated them to two tables on each secondary server as shown in Figure 4-3 on page 123. The sales table is split into two tables — one that contains sales for the eastern region for one of the secondary servers, while the other contains sales for the western region for the other secondary server. The DDL used in creating these tables is shown in Example 4-31 on page 146, Example 4-33 on page 147 and Example 4-32 on page 146.

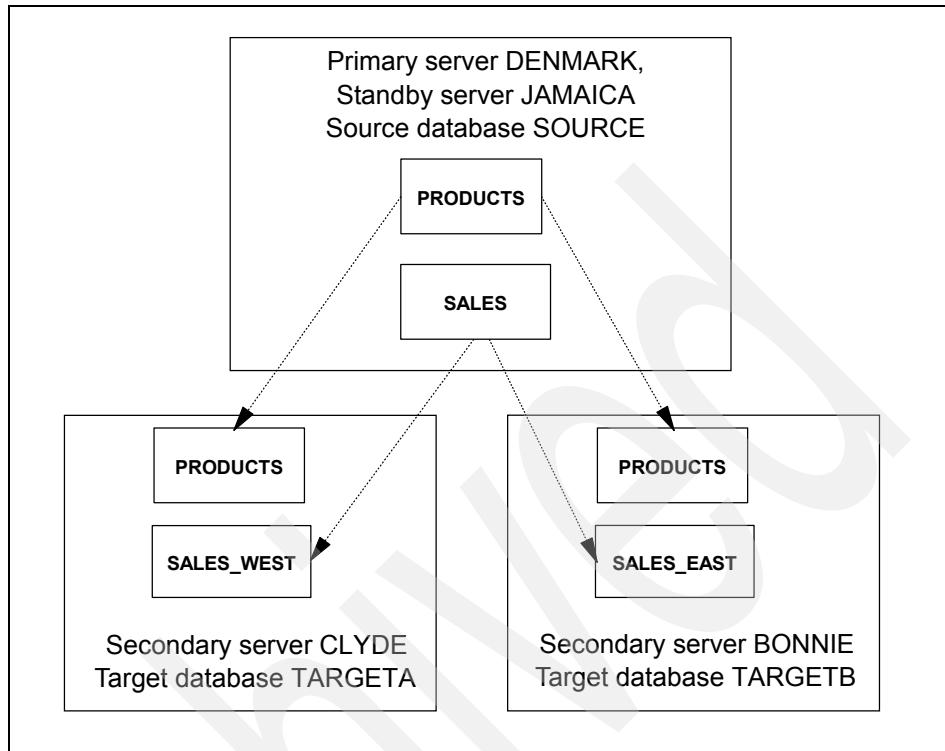


Figure 4-5 Tables used in the unidirectional replication scenario

4.5 Set up of the environment

Figure 4-6 provides an overview of the steps involved in setting up the HADR unidirectional Q replication environment described in Figure 4-1 on page 120 and Figure 4-5. The scripts used can be downloaded from the Redbooks Web site:

<ftp://www.redbooks.ibm.com/redbooks/SG247216/>

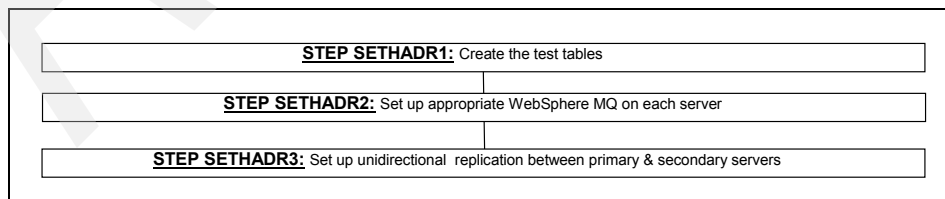


Figure 4-6 Overview of steps to set up the unidirectional scenario

The steps shown in Figure 4-5 on page 145 are described in the following sections.

4.5.1 STEP SETHADR1: Create the test tables

The test tables used in the scenario as described in Figure 4-5 on page 145 are created using DDL shown in Example 4-31, Example 4-33 on page 147, and Example 4-32.

Example 4-31 Create test tables on the primary server DENMARK

```
CREATE TABLE DB2INST9.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 DB2INST9.PRODUCT_PK ON
DB2INST9.PRODUCTS (PRODUCT_ID);
```

```
CREATE TABLE DB2INST9.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 DB2INST9.SALES_PK ON DB2INST9.SALES
(PRODUCT_ID, STORE_ID, DATE_OF_SALE);
```

Example 4-32 Create test tables on the secondary server CLYDE

```
CREATE TABLE DB2INST9.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 DB2INST9.PRODUCT_PK ON DB2INST9.PRODUCTS
(PRODUCT_ID);

CREATE TYPE 2 UNIQUE INDEX DB2INST9.SALES_EAST_PK ON DB2INST9.SALES_EAST
(PRODUCT_ID, STORE_ID, DATE_OF_SALE);

CREATE TABLE DB2INST9.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 DB2INST9.SALES_WEST_PK ON DB2INST9.SALES_WEST
(PRODUCT_ID, STORE_ID, DATE_OF_SALE);

```

Example 4-33 Create test tables on the secondary server BONNIE

```

CREATE TABLE DB2INST9.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 DB2INST9.PRODUCT_PK ON DB2INST9.PRODUCTS
(PRODUCT_ID);

CREATE TABLE DB2INST9.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 DB2INST9.SALES_EAST_PK ON DB2INST9.SALES_EAST
(PRODUCT_ID, STORE_ID, DATE_OF_SALE);
```

4.5.2 STEP SETHADR2: Set up appropriate WebSphere MQ

In this step, WebSphere MQ needs to be set up and configured on SEVERN, while WebSphere MQ clients need to be installed on the primary server DENMARK, standby server JAMAICA, and secondary servers CLYDE and BONNIE.

Note: It is assumed that WebSphere MQ V6 on SEVERN, and WebSphere MQ client on DENMARK, JAMAICA, CLYDE and BONNIE have been installed. For information about installing these products, refer to *WebSphere MQ for AIX V6.0 Quick Beginnings*, GC34-6478.

Example 4-34 shows the definition of the WebSphere MQ objects on SEVERN.

Example 4-35 on page 150 shows the starting of all the listener ports 1550, 1551 and 1552 on SEVERN.

Example 4-36 on page 150 through Example 4-39 on page 150 show the configuration of the WebSphere MQ Client to access SEVERN from CLYDE, BONNIE, DENMARK and JAMAICA respectively.

Example 4-34 Set up WebSphere MQ on MQ server SEVERN

1) Create the Queue Manager on severn

```
crtmqm -q QM_DWH
```

2) Start the Queue Manager on severn

```
strmqm QM_DWH
```

3) Start a WebSphere MQ listener

```
runmqtsr -m QM_DWH -t tcp -p 1550 &
```

```
runmqtsr -m QM_DWH -t tcp -p 1551 &
```

```
runmqtsr -m QM_DWH -t tcp -p 1552 &
```

4) Create the queue/channel objects

```
runmqsc QM_DWH <<!
```

```
DEFINE QLOCAL(ADMINQ) +
```

```
REPLACE +
```

```
DESCR('ADMINQ FOR SOURCE CAPTURE') +
```

```
PUT(ENABLED) +
```

```
GET(ENABLED) +
```

```
DEFPSIST(YES) +
```

```
MAXMSGL(4194403) +
```

```

MAXDEPTH(5000)

DEFINE QLOCAL(RESTARTQ) +
REPLACE +
DESCR('RESTARTQ FOR SOURCE CAPTURE') +
PUT(ENABLED) +
GET(ENABLED) +
DEFPSIST(YES) +
MAXMSGL(4194403) +
MAXDEPTH(5000)

DEFINE QMODEL(IBMQREP.SPILL.MODELQ) +
REPLACE +
DEFSOPT(SHARED) +
MAXDEPTH(500000) +
MSGDLVSQ(FIFO) +
DEFTYPE(PERMDYN) +
MAXMSGL(100000)

DEFINE QLOCAL(Q2TA) +
REPLACE +
DESCR('SEND RECEIVE QUEUE - SOURCE TO TARGETA') +
GET(ENABLED) +
SHARE +
DEFPSIST(YES) +
MSGDLVSQ(PRIORITY) +
MAXMSGL(4194403) +
MAXDEPTH(5000)

DEFINE QLOCAL(Q2TB) +
REPLACE +
DESCR('SEND RECEIVE QUEUE - SOURCE TO TARGETB') +
GET(ENABLED) +
SHARE +
DEFPSIST(YES) +
MSGDLVSQ(PRIORITY) +
MAXMSGL(4194403) +
MAXDEPTH(5000)

DEFINE CHANNEL(S2T) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER('mqm')
DEFINE CHANNEL(S2TA) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER('mqm')
DEFINE CHANNEL(S2TB) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER('mqm')

ALTER QMGR +
MAXUMSGS(10000) +
MAXMSGL(4194403)
!
# END OF SCRIPT

```

exit 0

Example 4-35 Start all the WebSphere MQ listener ports on SEVERN

```
runmqtsr -m QM_DWH -t TCP -p 1550
runmqtsr -m QM_DWH -t TCP -p 1551
runmqtsr -m QM_DWH -t TCP -p 1552
```

Example 4-36 Configure WebSphere MQ client on CLYDE

```
export MQSERVER="S2TA/TCP/9.43.86.59(1551)"
export ASNUSEMQCLIENT=TRUE
```

Example 4-37 Configure WebSphere MQ client on BONNIE

```
export MQSERVER="S2TB/TCP/9.43.86.59(1552)"
export ASNUSEMQCLIENT=TRUE
```

Example 4-38 Configure WebSphere MQ client on DENMARK

```
export MQSERVER="S2T/TCP/9.43.86.59(1550)"
export ASNUSEMQCLIENT=TRUE
```

Example 4-39 Configure WebSphere MQ client on JAMAICA

```
export MQSERVER="S2T/TCP/9.43.86.59(1550)"
export ASNUSEMQCLIENT=TRUE
```

4.5.3 STEP SETHADR3: Set up unidirectional replication

In this step, set up the Q replication objects for unidirectional replication as shown in Figure 4-7.

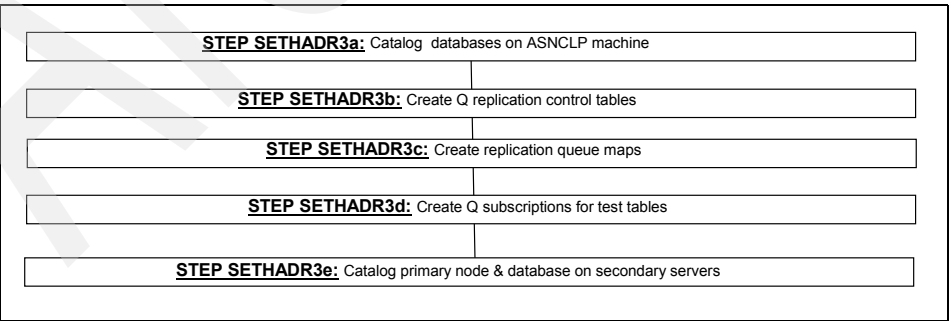


Figure 4-7 Overview of steps to set up the unidirectional replication environment

The steps shown in Figure 4-7 are described in the following sections.

STEP SETHADR3a: Catalog databases on ASNCLP machine

In this step, catalog the databases on the primary server DENMARK, and secondary servers CLYDE and BONNIE on the client machine as shown in Example 4-40. This is required for executing the ASNCLP commands in “STEP SETHADR3b: Create Q replication control tables” on page 151.

Example 4-40 Catalog databases on the ASNCLP machine

```
catalog tcpip node DENMARK remote 9.43.86.58 server 50000;  
catalog tcpip node CLYDE remote 9.43.86.56 server 50000;  
catalog tcpip node BONNIE remote 9.43.86.57 server 50000;  
catalog db SOURCE at node DENMARK;  
catalog db TARGETA at node CLYDE;  
catalog db TARGETB at node BONNIE;
```

STEP SETHADR3b: Create Q replication control tables

WebSphere II Q replication configuration may be performed using the Replication Center GUI, or by using ASNCLP commands from the Unix command line. This section documents the configuration of the unidirectional Q replication scenario using ASNCLP commands.

Note: ASNCLP commands are executed in the `asnclp` utility. After executing a command, the returned text should be reviewed very carefully since ASNCLP tends to be very wordy, even for a successful execution.

Important: Before running ASNCLP, the “qrepladm” user environment must be configured to provide support for Java™ as shown in Example 4-41.

Example 4-41 Set up Java environment for userid “qrepladm”

```
PATH=$PATH:/usr/java14/bin  
  
CP=/db2_data/db2inst9/sqllib/java/Common.jar  
CP=$CP:/db2_data/db2inst9/sqllib/tools/db2cmn.jar  
CP=$CP:/db2_data/db2inst9/sqllib/tools/db2replapis.jar  
CP=$CP:/db2_data/db2inst9/sqllib/tools/db2qreplapis.jar  
CP=$CP:/db2_data/db2inst9/sqllib/tools/jt400.jar  
CLASSPATH=$CLASSPATH:$CP
```

To ensure that the environment has been set up properly, run the **asnclp** command. If the “>Repl” command line appears, the set up was successful. Enter **quit** to exit the “Repl>” command line as shown in Example 4-42.

Example 4-42 asnclp set up configuration

```
$ asnclp
Repl >
Repl > quit
ASN1953I  ASNCLP : Command completed.
```

We recommend that the commands to perform the various tasks be stored in a file and then executed by directing the file to the **asnclp** utility as shown in Example 4-43.

Example 4-43 Execute commands from a file

```
asnclp -f <command_file>
## where <command_file> is the name of the file containing the asnclp commands
```

Example 4-44 lists the ASNCLP commands for creating the Q replication control tables on the primary server DENMARK, and secondary servers CLYDE and BONNIE.

Example 4-44 Create the Q replication control tables on

```
ASNCLP SESSION SET TO Q REPLICATION;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;

SET QMANAGER "QM_DWH" FOR CAPTURE SCHEMA;
SET QMANAGER "QM_DWH" FOR APPLY SCHEMA;

SET SERVER CAPTURE TO DB SOURCE ID db2inst9 PASSWORD "xxx";
SET CAPTURE SCHEMA SOURCE SOURCE;

CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "RESTARTQ" ADMINQ
"ADMINQ" ;

SET SERVER TARGET TO DB TARGETA ID db2inst9 PASSWORD "xxx";
SET APPLY SCHEMA TARGETA;

CREATE CONTROL TABLES FOR APPLY SERVER ;

SET SERVER TARGET TO DB TARGETB ID db2inst9 PASSWORD "xxx";
SET APPLY SCHEMA TARGETB;
```

STEP SETHADR3c: Create replication queue maps

Example 4-45 on page 153 lists the ASNCPL commands for creating the replication queue map pair on the primary server DENMARK and secondary server BONNIE, while Example 4-46 on page 153 lists the ASNCPL commands for creating the replication queue map pair on the primary server DENMARK and secondary server CLYDE.

Example 4-45 Create replication queue maps 1/2

```
ASNCPL SESSION SET TO Q REPLICATION;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;

SET SERVER CAPTURE TO DB SOURCE ID db2inst9 PASSWORD "xxx";
SET SERVER TARGET TO DB TARGETA ID db2inst9 PASSWORD "xxx";

SET QMANAGER "QM_DWH" FOR CAPTURE SCHEMA;
SET QMANAGER "QM_DWH" FOR APPLY SCHEMA;

SET CAPTURE SCHEMA SOURCE SOURCE;
SET APPLY SCHEMA TARGETA;

CREATE REPLQMAP S_TO_TA_MAP USING ADMINQ "ADMINQ" RECVQ "Q2TA" SENDQ "Q2TA"
HEARTBEAT INTERVAL 0;
```

Example 4-46 Create replication queue maps 2/2

```
ASNCPL SESSION SET TO Q REPLICATION;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;

SET SERVER CAPTURE TO DB SOURCE ID db2inst9 PASSWORD "xxx";
SET SERVER TARGET TO DB TARGETB ID db2inst9 PASSWORD "xxx";

SET QMANAGER "QM_DWH" FOR CAPTURE SCHEMA;
SET QMANAGER "QM_DWH" FOR APPLY SCHEMA;

SET CAPTURE SCHEMA SOURCE SOURCE;
SET APPLY SCHEMA TARGETB;

CREATE REPLQMAP S_TO_TB_MAP USING ADMINQ "ADMINQ" RECVQ "Q2TB" SENDQ "Q2TB"
HEARTBEAT INTERVAL 0;
```

STEP SETHADR3d: Create Q subscriptions for test tables

Example 4-47 lists the ASNCPL commands for creating the Q subscriptions for the PRODUCTS and SALES tables on the primary server DENMARK and

secondary server CLYDE, while Example 4-48 on page 154 lists the ASNCPLP commands for creating the Q subscriptions for the PRODUCTS and SALES tables on the primary server DENMARK and secondary server BONNIE.

Example 4-47 Create Q subscriptions for the test tables 1/2

```
ASNCPLP SESSION SET TO Q REPLICATION;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
#SET RUN SCRIPT LATER;

SET SERVER CAPTURE TO DB SOURCE ID db2inst9 PASSWORD "xxx";
SET SERVER TARGET TO DB TARGETA ID db2inst9 PASSWORD "xxx";

SET QMANAGER "QM_DWH" FOR CAPTURE SCHEMA;
SET QMANAGER "QM_DWH" FOR APPLY SCHEMA;

SET CAPTURE SCHEMA SOURCE SOURCE;
SET APPLY SCHEMA TARGETA;

CREATE QSUB USING REPLQMAP S_TO_TA_MAP
(SUBNAME SUBA-PRODUCTS
 DB2INST9.PRODUCTS
 OPTIONS
 HAS LOAD PHASE I SPILL_MODELQ "IBMQREP.SPILL.MODELQ"
 EXIST TARGET NAME DB2INST9.PRODUCTS
 LOAD TYPE 3 );

CREATE QSUB USING REPLQMAP S_TO_TA_MAP
(SUBNAME SUBA-SALES
 DB2INST9.SALES
 OPTIONS
 SEARCH CONDITION "WHERE :REGION_CODE='EAST'"
 ALL CHANGED ROWS Y
 HAS LOAD PHASE I SPILL_MODELQ "IBMQREP.SPILL.MODELQ"
 EXIST TARGET NAME DB2INST9.SALES_EAST
 TRGCOLS INCLUDE(PRODUCT_ID,STORE_ID,QUANTITY_SOLD,DATE_OF_SALE)
 LOAD TYPE 3 );
```

Example 4-48 Create Q subscriptions for the test tables 2/2

```
ASNCPLP SESSION SET TO Q REPLICATION;
SET RUN SCRIPT NOW STOP ON SQL ERROR ON;

SET SERVER CAPTURE TO DB SOURCE ID db2inst9 PASSWORD "xxx";
SET SERVER TARGET TO DB TARGETB ID db2inst9 PASSWORD "xxx";

SET QMANAGER "QM_DWH" FOR CAPTURE SCHEMA;
SET QMANAGER "QM_DWH" FOR APPLY SCHEMA;
```

```

SET CAPTURE SCHEMA SOURCE SOURCE;
SET APPLY SCHEMA TARGETB;

CREATE QSUB USING REPLQMAP S_TO_TB_MAP
  (SUBNAME SUBB-PRODUCTS
   DB2INST9.PRODUCTS
   OPTIONS
   HAS LOAD PHASE I SPILL_MODELQ "IBMQREP.SPILL.MODELQ"
   EXIST TARGET NAME DB2INST9.PRODUCTS
   LOAD TYPE 3 );

CREATE QSUB USING REPLQMAP S_TO_TB_MAP
  (SUBNAME SUBB-SALES
   DB2INST9.SALES
   OPTIONS
   SEARCH CONDITION "WHERE :REGION_CODE='WEST'"
   ALL CHANGED ROWS Y
   HAS LOAD PHASE I SPILL_MODELQ "IBMQREP.SPILL.MODELQ"
   EXIST TARGET NAME DB2INST9.SALES_WEST
   TRGCOLS INCLUDE(PRODUCT_ID,STORE_ID,QUANTITY_SOLD,DATE_OF_SALE)
   LOAD TYPE 3 );

```

STEP SETHADR3e: Catalog primary node and database

In this step, catalog the primary server DENMARK node and SOURCE database on the secondary servers CLYDE and BONNIE as shown in Example 4-49 and Example 4-50 respectively. This is required to support automatic load of data from the primary server DENMARK.

Example 4-49 Catalog the primary server node and database on the secondary server CLYDE

```

catalog tcpip node DENMARK remote 9.43.86.58 server 50000;
catalog db SOURCE at node DENMARK;

```

Example 4-50 Catalog the primary server node and database on the secondary server BONNIE

```

catalog tcpip node DENMARK remote 9.43.86.58 server 50000;
catalog db SOURCE at node DENMARK;

```

STEP SETHADR3f: Create password file on secondary servers

Since automatic loading of the tables will be used in our test cases, a password file (*asnpwd.aut*) needs to be created in the APPLY_PATH of the target system so that the Q Apply program can access the source database and read the source tables. Example 4-51 creates the password file for userid “qrepladm” on

node CLYDE to access the source tables on DENMARK, while Example 4-52 does the same on node BONNIE.

Example 4-51 Create the password file on the secondary server CLYDE

```
cd ~qrepladm/apply
asnpwd init
asnpwd add alias SOURCE id db2inst9 password xxx
```

Example 4-52 Create the password file on the secondary server BONNIE

```
cd ~qrepladm/apply
asnpwd init
asnpwd add alias SOURCE id db2inst9 password xxx
```

4.6 Planned takeover

A planned takeover of the primary server by a standby server 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 is typically not extended in nature.

The following subsections describe the recommended procedure for a planned takeover as shown in Figure 4-8 on page 157 and include steps to verify that the procedure ensures that unidirectional replication continues seamlessly without any data loss.

The assumptions made about this procedure are as follows:

- ▶ HADR is up and running with an application workload accessing the primary server DENMARK.
- ▶ DB2 Client Reroute is in effect and alternate server information is updated on the client machine.
- ▶ Application workload is not explicitly halted throughout the procedure — however, the client may experience some unavailability during the switchover of the primary to the standby server.
- ▶ Unidirectional replication has been up and running, but Q Capture is not started on the primary server DENMARK, and Q Apply is not started on the secondary servers CLYDE and BONNIE.

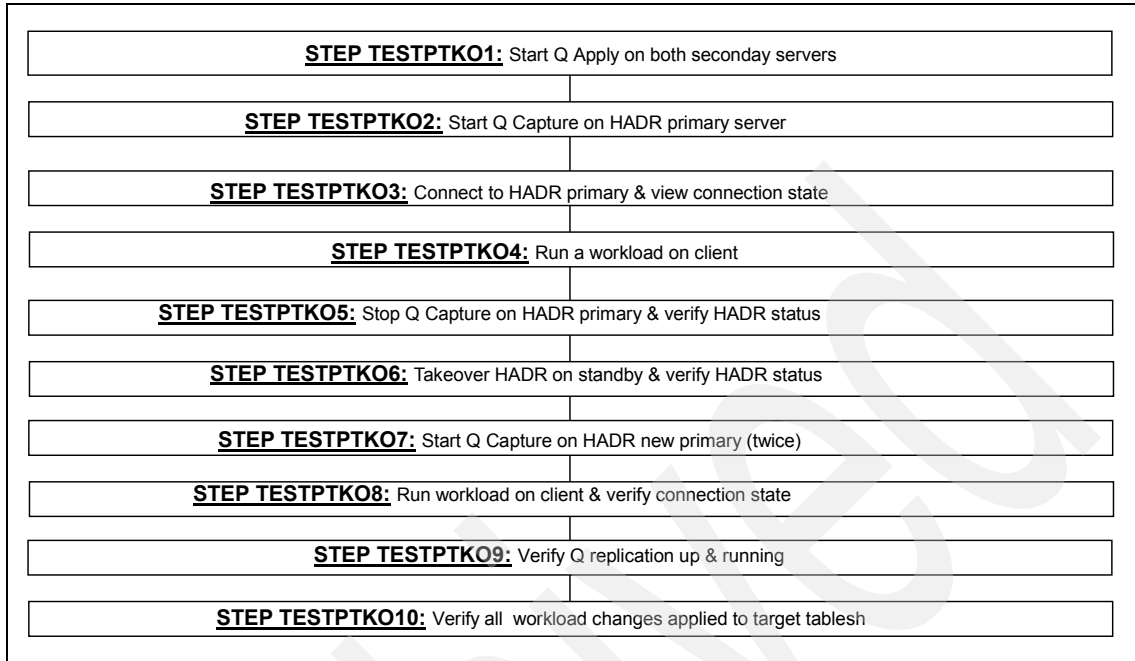


Figure 4-8 Overview of planned takeover steps

The steps shown in Figure 4-8 are described briefly in the following sections.

4.6.1 STEP TESTPTKO1: Start Q Apply on both secondary servers

Start Q Apply on the secondary servers CLYDE and BONNIE by executing the commands shown in Example 4-53, and Example 4-55 on page 158 respectively, the outputs of which indicate that the Q Apply program started processing the receive queue. You can monitor the contents of the Q Apply log on CLYDE (as shown in Example 4-54 on page 158) and BONNIE (as shown in Example 4-56 on page 158) which show the Q Apply program initializing successfully and the various agents ready to process the receive queue.

The status of the Q Apply program may also be viewed by issuing the **asnqacmd** command which displays the state of the various threads associated with the Q Apply process (not shown here).

Example 4-53 Start Q Apply on the secondary server CLYDE

```
export MQSERVER="S2TA/TCP/9.43.86.59(1551)"
export ASNUSEMQCLIENT=TRUE

asnqapp apply_server=TARGETA apply_schema=TARGETA apply_path=/DB2
```


4.6.2 STEP TESTPTKO2: Start Q Capture on HADR primary

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 (not shown here).

Example 4-58 Start Q Capture log contents on the primary server DENMARK

```

2006-03-07-11.05.55.026685 <setEnvDprRIB> ASN0600I  "Q Capture" :  "" :
"Initial" : Program "mqpub 9.1.0" is starting.
.....
2006-03-07-11.05.57.072263 <asnParmClass::printParms> ASN0529I  "Q Capture" :
"SOURCE" : "Initial" : The value of "STARTMODE" was set to "WARMSI" at startup
by the following method: "PARAMETERS TABLE".
.....
2006-03-07-11.05.57.164600 <subMgr::processSubscriptions> ASN7000I  "Q Capture"
: "SOURCE" : "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.
.....
2006-03-07-11.05.57.170278 <asnqwk> ASN0572I  "Q Capture" : "SOURCE" :
"WorkerThread" : The program initialized successfully.

```

4.6.3 STEP TESTPTKO3: Connect to HADR primary and view state

In this step, connect to the SOURCE database on the HADR primary server DENMARK from the client machine via the **get connection state** command, and confirm that the client is accessing DENMARK as shown in Example 4-59 and Example 4-60. Additionally, using the **LIST DB DIRECTORY** command, verify that the Alternate server hostname information has been updated to reflect the standby server JAMAICA as shown in Example 4-61 on page 161.

Note: This step would typically not be part of a “real-world” planned takeover procedure — it is included here for you to better understand the state transitions that occur in a planned takeover scenario.

Example 4-59 Connect to the primary server DENMARK from the client machine

```
DB2 CONNECT TO source USER db2inst9 USING xxx;
```

```

Database Connection Information
Database server      = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = SOURCE

```

Example 4-60 Connection state from the client machine

```
DB2 GET CONNECTION STATE
```

```

Database Connection State
Connection state      = Connectable and Connected
Connection mode       = SHARE
Local database alias  = SOURCE
Database name         = SOURCE

```

```
Hostname           = denmark
Service name       = 50000
```

Example 4-61 List database directory from the client machine

DB2 LIST DB DIRECTORY

System Database Directory

Number of entries in the directory = 4

Database 1 entry:

```
Database alias      = SOURCE
Database name       = SOURCE
Node name           = DENMARK
Database release level = a.00
Comment             =
Directory entry type = Remote
Catalog database partition number = -1
Alternate server hostname = jamaica
Alternate server port number = 50000
.....
```

4.6.4 STEP TESTPTKO4: Run workload on client

In this step we simulate a “real-world” environment by running an application workload that updates the PRODUCTS and SALES tables on the SOURCE database on the primary server from the client machine as shown in Example 4-62. Later on in this procedure, we will ensure that the changes introduced by this workload is reflected in the secondary servers CLYDE and BONNIE.

Example 4-62 Run a workload from the client machine

```
DB2 +c -tvf w1.sql
--Contents of w1.sql
DELETE FROM DB2INST9.PRODUCTS;
DELETE FROM DB2INST9.SALES;

INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (10, 'WIDGETS', 'TOOLS', 1.00, 100);
INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (20, 'THINGAMABOBBER', 'TOOLS', 5.00, 100);
INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (30, 'WHATCHAMACALLITS', 'TOOLS', 10.00, 100);
```

```

INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (40, 'DOODADS', 'TOOLS', 3.00, 100);
INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (50, 'THINGAMAGIGS', 'TOOLS', 2.00, 100);

INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (10, 100, 1000, '2006-03-01', 'EAST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (20, 120, 2000, '2006-03-01', 'EAST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (30, 100, 5000, '2006-03-02', 'EAST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (40, 140, 4000, '2006-03-02', 'EAST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (50, 150, 5000, '2006-03-02', 'EAST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (30, 200, 5000, '2006-03-01', 'WEST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (40, 240, 4000, '2006-03-01', 'WEST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (50, 250, 5000, '2006-03-01', 'WEST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (10, 200, 1000, '2006-03-02', 'WEST');
INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (20, 220, 2000, '2006-03-02', 'WEST');

COMMIT;

```

4.6.5 STEP TESTPTKO5: Stop Q Capture on HADR primary and verify the HADR status

In this step, stop Q Capture on the primary server DENMARK by executing the commands shown in Example 4-63 on page 163 — stdout of Q Capture is shown

in Example 4-64. You can monitor the contents of the Q Capture log which shows the program as having stopped as shown in Example 4-65.

Note: It is desirable but not essential to stop the workload on the HADR primary server DENMARK at this point by stopping DB2 also. If the workload is not stopped, access to the primary server DENMARK will fail after takeover and DB2 Client Reroute will automatically switch the workload over to the standby server JAMAICA.

Example 4-63 Stop Q Capture on the HADR primary server DENMARK

asnqccmd capture_server=SOURCE capture_schema=SOURCE stop

2006-03-07-11.12.04.565498 ASN0522I "AsnQCcmd" : "SOURCE" : "Initial" : The program received the "STOP" command.

Example 4-64 Stop Q Capture stdout on the HADR primary server DENMARK

2006-03-07-11.05.57.164600 ASN7000I "Q Capture" : "SOURCE" : "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.

2006-03-07-11.05.57.170278 ASN0572I "Q Capture" : "SOURCE" : "WorkerThread" : The program initialized successfully.

2006-03-07-11.12.08.408936 ASN0573I "Q Capture" : "SOURCE" : "Initial" : The program was stopped.

[asnqwk] Leaving the asnqwk thread:

[asnqwk] total database transactions captured = 1

[asnqwk] total MQSeries transactions = 1

[asnqwk] total DB transactions published = 1

[asnqwk] total time WAITING for logrd = 366558 seconds

[asntxrd] Leaving logtxrd reader thread:

[asntxrd] 0 message(s) in notification queue

[asntxrd] transmgr memory in use = 0 bytes

[asntxrd] totalLogReadTime = 0.072 second(s)

[asntxrd] total logrd SLEEP time = 370.000 second(s)

[asntxrd] totalLogRecsRead = 30

[asntxrd] Queued a thread Termination Notification.

[waitForLogrdEnd] The logrd thread terminated with rc: 0

Example 4-65 Stop Q Capture log contents on the HADR primary server DENMARK

.....

2006-03-07-11.12.03.775369 <asnqwk> ASN7109I "Q Capture" : "SOURCE" : "WorkerThread" : At program termination, the highest log sequence number of a successfully processed transaction is "440D:BE81:0000:0001:0000" and the lowest

```
log sequence number of a transaction still to be committed is
"0000:0000:0000:122F:FD3C".
.....
2006-03-07-11.12.08.408936 <asnqcap::main> ASN0573I  "Q Capture" :  "SOURCE" :
"Initial" : The program was stopped.
```

4.6.6 STEP TESTPTKO6: Takeover HADR on standby and verify HADR status

Example 4-66 and Example 4-67 on page 165 show the HADR status of the primary server and standby server just prior to issuing the TAKEOVER command. The **db2pd** command issued on the primary server DENMARK (Example 4-66) shows the HADR Role to be “Primary”, State to be “Peer” and the SyncMode to be “Sync”. The **db2pd** command issued on the standby server JAMAICA (Example 4-67 on page 165) shows the HADR Role to be “Standby”, State to be “Peer” and the SyncMode to be “Sync”. This corresponds to a normal HADR operating normal environment.

Next, issue the TAKEOVER command on the standby server JAMAICA to switch the primary role from DENMARK as shown in Example 4-68 on page 165. The db2diag.log files on the primary server DENMARK (Example 4-69 on page 166) and the standby server JAMAICA (Example 4-70 on page 168) document in detail the progress of the takeover process, where the standby server JAMAICA becomes the primary server, and the primary server DENMARK becomes the standby server — a switch of roles.

Example 4-71 on page 170 and Example 4-72 on page 171 show the HADR status of the primary server and standby server after takeover processing has completed. The **db2pd** command issued on the now standby server DENMARK (Example 4-71 on page 170) shows the HADR Role to be “Standby”, State to be “Peer” and the SyncMode to be “Sync”. The **db2pd** command issued on the new primary server JAMAICA (Example 4-72 on page 171) shows the HADR Role to be “Primary”, State to be “Peer” and the SyncMode to be “Sync”. This corresponds to a planned takeover HADR operating environment.

Example 4-66 Check HADR status on the primary server DENMARK

```
db2pd -db source -hadr
```

```
Database Partition 0 -- Database SOURCE -- Active -- Up 0 days 02:38:49
```

```
HADR Information:
```

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer | Sync | 0 | 0 |

| | | |
|---------------|--------------------------------------|---------|
| ConnectStatus | ConnectTime | Timeout |
| Connected | Tue Mar 7 10:22:36 2006 (1141748556) | 120 |

| | |
|-----------|--------------|
| LocalHost | LocalService |
| denmark | 60000 |

| | | |
|-------------------|----------------------|-----------------------|
| RemoteHost | RemoteService | RemoteInstance |
| jamaica | 60001 | db2inst9 |

| | | |
|--------------|-----------|--------------------|
| PrimaryFile | PrimaryPg | PrimaryLSN |
| S0000071.LOG | 495 | 0x00000000122FFF64 |

| | | |
|--------------|-----------|--------------------|
| StandByFile | StandByPg | StandByLSN |
| S0000071.LOG | 495 | 0x00000000122FFF64 |

Example 4-67 Check HADR status on the standby server JAMAICA

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 3 days 21:07:05

HADR Information:

| | | | | |
|---------------------|--------------|-----------------|-------------------------|-----------------------------|
| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
| Standby Peer | | Sync | 0 | 0 |

| | | |
|---------------|--------------------------------------|---------|
| ConnectStatus | ConnectTime | Timeout |
| Connected | Tue Mar 7 10:22:36 2006 (1141748556) | 120 |

| | |
|------------------|---------------------|
| LocalHost | LocalService |
| jamaica | 60001 |

| | | |
|-------------------|----------------------|-----------------------|
| RemoteHost | RemoteService | RemoteInstance |
| denmark | 60000 | db2inst9 |

| | | |
|--------------|-----------|--------------------|
| PrimaryFile | PrimaryPg | PrimaryLSN |
| S0000071.LOG | 495 | 0x00000000122FFF64 |

| | | |
|--------------|-----------|--------------------|
| StandByFile | StandByPg | StandByLSN |
| S0000071.LOG | 495 | 0x00000000122FFF64 |

Example 4-68 Issue Takeover HADR on database source command on JAMAICA

db2 TAKEOVER HADR ON DATABASE SOURCE

DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.

Example 4-69 Contents of the db2diag.log on the primary server DENMARK

2006-03-07-13.02.43.926414-360 I2827A333 LEVEL: Warning
PID : 1228922 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrPD takeover,
probe:43006
MESSAGE : Info: Primary has started non-forced takeover request.

2006-03-07-13.02.43.927263-360 I3161A322 LEVEL: Warning
PID : 1228922 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrPD takeover,
probe:43002
MESSAGE : Info: Primary is switching to standby role.

2006-03-07-13.02.43.927876-360 E3484A400 LEVEL: Event
PID : 1540140 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-267
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSwitchDbFromRuntimeToStandby, probe:50122
CHANGE : Current log position at time of Role Switch - LSN = 00000000122FFF65

2006-03-07-13.02.43.951191-360 E3885A352 LEVEL: Event
PID : 1540140 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-267
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to S-Peer (was P-Peer)

2006-03-07-13.02.43.951440-360 I4238A330 LEVEL: Warning
PID : 1228922 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrPD takeover,
probe:43003
MESSAGE : Info: Primary has completed takeover (now standby).

2006-03-07-13.02.43.952360-360 I4569A315 LEVEL: Warning
PID : 1228922 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster,
probe:21251
MESSAGE : Info: Replaymaster Starting...

2006-03-07-13.02.43.952705-360 I4885A345 LEVEL: Warning

PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
 probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-03-07-13.02.43.953178-360 I5231A306 LEVEL: Severe
 PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, base sys utilities, sqlsrsu, probe:999
 MESSAGE : free tran stuff

2006-03-07-13.02.43.953535-360 I5538A332 LEVEL: Warning
 PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:300
MESSAGE : Starting Replay Master on standby.

2006-03-07-13.02.43.953725-360 I5871A317 LEVEL: Warning
 PID : 1228922 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster,
 probe:21252
MESSAGE : Info: Replaymaster request done.

2006-03-07-13.02.43.953848-360 I6189A295 LEVEL: Warning
 PID : 1228922 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20302
MESSAGE : Info: Primary Finished.

2006-03-07-13.02.43.953951-360 I6485A294 LEVEL: Warning
 PID : 1228922 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20341
MESSAGE : Info: Standby Started.

2006-03-07-13.02.43.954092-360 I6780A312 LEVEL: Warning
 PID : 1228922 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSPPrepareLogWrite,
 probe:10260
 MESSAGE : RCUStartLsn 00000000122FFF65

2006-03-07-13.02.44.901572-360 E7093A348 LEVEL: Warning
 PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0

INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:920
MESSAGE : ADM1602W Rollforward recovery has been initiated.

2006-03-07-13.02.44.901838-360 E7442A391 LEVEL: Warning
 PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:1740
MESSAGE : ADM1603I DB2 is invoking the forward phase of the database rollforward recovery.

2006-03-07-13.02.44.902068-360 I7834A419 LEVEL: Warning
 PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:710
 DATA #1 : <preformatted>
 Invoking database rollforward forward recovery,
 lowtranlsn 00000000122FFF65 minbufflsn 00000000122BEA19

2006-03-07-13.02.44.928969-360 I8254A374 LEVEL: Warning
 PID : 1683692 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
 DATA #1 : <preformatted>
 Using parallel recovery with 5 agents 7 QSets 28 queues and 2 chunks

2006-03-07-13.02.44.929405-360 E8629A370 LEVEL: Warning
 PID : 1540140 TID : 1 PROC : db2redom (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-268
 FUNCTION: DB2 UDB, recovery manager, sqlpParallelRecovery, probe:880
 DATA #1 : <preformatted>
 Resetting max shredder memory to 1924984 from 2867200

Example 4-70 Contents of the db2diag.log on the standby server JAMAICA

2006-03-07-13.02.43.785909-360 I4871A374 LEVEL: Warning
 PID : 495856 TID : 1 PROC : db2agent (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 APPHDL : 0-229 APPID: *LOCAL.db2inst9.060307190243
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
 probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-03-07-13.02.43.826315-360 I5246A318 LEVEL: Warning

PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
probe:47001

MESSAGE : Info: Standby has initiated a takeover.

2006-03-07-13.02.43.952123-360 I5565A320 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
probe:47002

MESSAGE : Info: Standby switching roles to primary.

2006-03-07-13.02.43.977487-360 I5886A395 LEVEL: Info
PID : 491544 TID : 1 PROC : db2redom (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-46
FUNCTION: DB2 UDB, recovery manager, sqlpRecReadLog, probe:4630
DATA #1 : <preformatted>
nextLsn 00000000122FFF65 rfwrd_ReadLsn 00000000122FFF64 ExtNum 72 firstLsn
0000124F8000

2006-03-07-13.02.43.978682-360 I6282A340 LEVEL: Warning
PID : 811020 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-46
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:1990
DATA #1 : <preformatted>
nextLsn 00000000122FFF65

2006-03-07-13.02.44.129720-360 E6623A387 LEVEL: Warning
PID : 811020 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-46
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:3600
**MESSAGE : ADM1605I DB2 is invoking the backward phase of database rollforward
recovery.**

2006-03-07-13.02.44.130058-360 I7011A390 LEVEL: Warning
PID : 811020 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-46
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:2210
DATA #1 : <preformatted>
Invoking database rollforward backward recovery, nextLsn: 00000000122FFF65

2006-03-07-13.02.44.513245-360 E7402A358 LEVEL: Warning
PID : 811020 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE

APPHDL : 0-46
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:6600
MESSAGE : ADM1611W The rollforward recovery phase has been completed.

2006-03-07-13.02.44.518474-360 I7761A332 LEVEL: Warning
 PID : 811020 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-46
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:9500
MESSAGE : Stopping Replay Master on standby.

2006-03-07-13.02.44.765299-360 E8094A314 LEVEL: Event
 PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
 probe:10000
CHANGE : HADR state set to P-Peer (was S-Peer)

2006-03-07-13.02.44.885189-360 I8409A330 LEVEL: Warning
 PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
 probe:47003
MESSAGE : Info: Standby has completed takeover (now primary).

2006-03-07-13.02.44.885828-360 I8740A295 LEVEL: Warning
 PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20342
MESSAGE : Info: Standby Finished.

2006-03-07-13.02.45.005034-360 I9036A294 LEVEL: Warning
 PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20301
MESSAGE : Info: Primary Started.

Example 4-71 Check HADR status on the primary server DENMARK

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 0 days 02:41:02

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Standby | Peer | Sync | 0 | 0 |

| | | |
|---------------|-------------|---------|
| ConnectStatus | ConnectTime | Timeout |
|---------------|-------------|---------|

Connected Tue Mar 7 10:22:36 2006 (1141748556) 120

| | | | |
|------------------------------|-----------|-------------------------------|-----------------------------------|
| LocalHost denmark | | LocalService 60000 | |
| RemoteHost jamaica | | RemoteService 60001 | RemoteInstance db2inst9 |
| PrimaryFile | PrimaryPg | PrimaryLSN | |
| S0000071.LOG | 495 | 0x00000000122FFF64 | |
| StandByFile | StandByPg | StandByLSN | |
| S0000071.LOG | 495 | 0x00000000122FFF64 | |

Example 4-72 Check HADR status on the standby server JAMAICA

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 3 days 21:08:59

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer | Sync | 0 | 0 |

| | | |
|---------------|--------------------------------------|---------|
| ConnectStatus | ConnectTime | Timeout |
| Connected | Tue Mar 7 10:22:36 2006 (1141748556) | 120 |

| | | | |
|------------------------------|-----------|-------------------------------|-----------------------------------|
| LocalHost jamaica | | LocalService 60001 | |
| RemoteHost denmark | | RemoteService 60000 | RemoteInstance db2inst9 |
| PrimaryFile | PrimaryPg | PrimaryLSN | |
| S0000071.LOG | 495 | 0x00000000122FFF64 | |
| StandByFile | StandByPg | StandByLSN | |
| S0000071.LOG | 495 | 0x00000000122FFF64 | |

4.6.7 STEP TESTPTK07: Start the Q Capture on the new HADR primary (twice)

In this step, Q Capture must be started *twice* on the HADR new primary server JAMAICA because the first attempt fails with an error.

Start Q Capture on the HADR new primary server JAMAICA by executing the appropriate **asnlccmd** command which fails with an error and does not start as shown in Example 4-73. You can monitor the contents of the Q Capture log which shows the program as having encountered an unexpected error and stopping as shown in Example 4-74.

Restarting Q Capture again by executing the appropriate **asnlccmd** command on the HADR new primary server JAMAICA results in a successful start as shown in Example 4-75 on page 173 the output of which indicates that the Q Capture program initialized successfully. You can monitor the contents of the Q Capture log which shows the Q Capture program initializing successfully as shown in Example 4-76 on page 173.

Example 4-73 Start Q Capture on the new primary server JAMAICA

```
export MQSERVER="S2T/TCP/9.43.86.59(1550)"
export ASNUSEMQCLIENT=TRUE
```

```
asnlcap capture_server=SOURCE capture_schema=SOURCE capture_path=/DB2
```

```
2006-03-07-13.03.43.139672 ASN0122E CAPTURE "SOURCE" : "Initial". An error
occurred while reading the restart information or DB2 log. The Capture program
will terminate.
```

```
2006-03-07-13.03.44.171646 ASN0573I "Q Capture" : "SOURCE" : "Initial" : The
program was stopped.
```

Example 4-74 Start Q Capture log contents on the new primary server JAMAICA

```
.....
```

```
2006-03-07-13.03.43.108536 <asnParmClass::printParms> ASN0529I "Q Capture" :
"SOURCE" : "Initial" : The value of "STARTMODE" was set to "WARMSI" at startup
by the following method: "PARAMETERS TABLE".
```

```
.....
```

```
2006-03-07-13.03.43.125932 <logrd::validateWarmStart> ASN8999D "Q Capture" :
"SOURCE" : "Initial" : Capture has detected the source database has been
restored/rollforward
```

```
2006-03-07-13.03.43.139621 <asnlcap::main> ASN0589I "Q Capture" : "SOURCE" :
"Initial" The program received an unexpected return code "924" from routine
"doWarmStart".
```

```
2006-03-07-13.03.43.139672 <asnlcap::main> ASN0122E CAPTURE "SOURCE" :
"Initial". An error occurred while reading the restart information or DB2 log.
The Capture program will terminate.
```

```
.....
```

```
2006-03-07-13.03.43.139927 <threadPrologue> ASN0589I "Q Capture" : "SOURCE" :
"AdminThread" The program received an unexpected return code "2009" from
routine "setRecoveryPoint".
```

```
2006-03-07-13.03.43.139930 <erWhatSignal> ASN0591I "Q Capture" : "SOURCE" :
"PruneThread" The thread "PruneThread" received "Handled" signal "SIGUSR1".
```

```

2006-03-07-13.03.43.139974 <asnmqadmt> ASN0589I  "Q Capture" : "SOURCE" :
"AdminThread" The program received an unexpected return code "2009" from
routine "threadPrologue".
2006-03-07-13.03.43.140004 <threadPrologue> ASN0589I  "Q Capture" : "SOURCE" :
"PruneThread" The program received an unexpected return code "2009" from
routine "setRecoveryPoint".
2006-03-07-13.03.43.140027 <asnmqprun> ASN0589I  "Q Capture" : "SOURCE" :
"PruneThread" The program received an unexpected return code "2009" from
routine "threadPrologue".
2006-03-07-13.03.44.171646 <asnmqcap::main> ASN0573I  "Q Capture" : "SOURCE" :
"Initial" : The program was stopped.

```

Example 4-75 Start Q Capture on the new primary server JAMAICA

```

export MQSERVER="S2T/TCP/9.43.86.59(1550)"
export ASNUSEMQCLIENT=TRUE

asnmqcap capture_server=SOURCE capture_schema=SOURCE capture_path=/DB2

2006-03-07-13.09.24.553523 ASN7000I  "Q Capture" : "SOURCE" : "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.
2006-03-07-13.09.24.559033 ASN0572I  "Q Capture" : "SOURCE" : "WorkerThread" :
The program initialized successfully.

```

Example 4-76 Start Q Capture log contents on the new primary server JAMAICA

```

.....
2006-03-07-13.09.22.397378 <setEnvDprRIB> ASN0600I  "Q Capture" : "" :
"Initial" : Program "mqpub 9.1.0" is starting.
.....
2006-03-07-13.09.24.420514 <asnmParmClass::printParms> ASN0529I  "Q Capture" :
"SOURCE" : "Initial" : The value of "STARTMODE" was set to "WARMSI" at startup
by the following method: "PARAMETERS TABLE".
.....
2006-03-07-13.09.24.553523 <subMgr::processSubscriptions> ASN7000I  "Q Capture"
: "SOURCE" : "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.
2006-03-07-13.09.24.558982 <waitForLogrdInit> ASN7108I  "Q Capture" : "SOURCE"
: "WorkerThread" : At program initialization, the highest log sequence number
of a successfully processed transaction is "440D:BE81:0000:0001:0000" and the
lowest log sequence number of a transaction still to be committed is
"0000:0000:0000:122C:F26A".
2006-03-07-13.09.24.559033 <asnmqwk> ASN0572I  "Q Capture" : "SOURCE" :
"WorkerThread" : The program initialized successfully.

```

4.6.8 STEP TESTPTK08: Run workload on client and verify state

In this step, run another workload on the client machine as shown in Example 4-77, the output of which is shown in Example 4-78 on page 175. Next review the connection state on the client machine as shown in Example 4-82 on page 179.

Note: In a “real-world” environment, the application workload is probably running continuously from a number of client machines.

The output of the workload shown in Example 4-78 on page 175 shows that the first SQL statement “UPDATE DB2INST9.PRODUCTS SET PRICE = 8.00 WHERE PRODUCT_ID = 20;” encounters a problem connecting to the old primary server DENMARK, but DB2 Client Reroute automatically reroutes the query to the alternate server JAMAICA and succeeds. Message SQL301018N in Example 4-78 on page 175 indicates this occurrence. Example 4-79 on page 176 shows the content of the db2diag.log on the client machine indicating the successful initialization and completion of DB2 Client Reroute.

Example 4-80 on page 178 shows the **get connection state** command issued from the client machine indicating that the client is now connected to the new primary server JAMAICA.

Example 4-82 on page 179 shows the **LIST DB DIRECTORY** command issued from the client machine indicating that the Alternate server hostname has been changed to DENMARK from JAMAICA because the roles have been switched.

Example 4-77 Run a workload on the client machine 1/2

```
DB2 +c -tvf w2.sql
--Contents of w2.sql
UPDATE DB2INST9.PRODUCTS SET PRICE = 8.00 WHERE PRODUCT_ID = 20;

INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (60, 'HEAVY HAMMER', 'TOOLS', 100.00, 100);

UPDATE DB2INST9.PRODUCTS SET PRICE = 8.00 WHERE PRODUCT_ID = 20;

DELETE FROM DB2INST9.PRODUCTS WHERE PRODUCT_ID = 50;

INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (60, 160, 6000, '2006-03-03', 'EAST');

DELETE FROM DB2INST9.SALES WHERE PRODUCT_ID = 50;
UPDATE DB2INST9.PRODUCTS SET MANUFACT_ID = 200 WHERE PRODUCT_ID = 20 ;
```

```

INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (30, 200, 3000, '2006-03-03', 'WEST');

UPDATE DB2INST9.SALES SET REGION_CODE = 'WEST'
WHERE PRODUCT_ID = 40 AND STORE_ID = 140 AND DATE_OF_SALE = '2006-03-02';

COMMIT;

```

Example 4-78 Run a workload on the client machine 2/2

```

UPDATE DB2INST9.PRODUCTS SET PRICE = 8.00 WHERE PRODUCT_ID = 20

```

DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned: **SQL30108N** A connection failed but has been re-established. The hostname or IP address is "jamaica" and the service name or port number is "50000". Special registers may or may not be re-attempted (Reason code = "1"). **SQLSTATE=08506**

```

INSERT INTO DB2INST9.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 DB2INST9.PRODUCTS SET PRICE = 8.00 WHERE PRODUCT_ID = 20

```

DB20000I The SQL command completed successfully.

```

DELETE FROM DB2INST9.PRODUCTS WHERE PRODUCT_ID = 50

```

DB20000I The SQL command completed successfully.

```

INSERT INTO DB2INST9.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 DB2INST9.SALES WHERE PRODUCT_ID = 50

```

DB20000I The SQL command completed successfully.

```

UPDATE DB2INST9.PRODUCTS SET MANUFACT_ID = 200 WHERE PRODUCT_ID = 20

```

DB20000I The SQL command completed successfully.

```

INSERT INTO DB2INST9.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 DB2INST9.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.

COMMIT

DB20000I The SQL command completed successfully.

Example 4-79 db2diag.log contents on the client machine

2006-03-07-13.07.25.452000-480 I380129H1019 LEVEL: Warning
PID : 2416 TID : 2392 PROC : db2bp.exe
INSTANCE: DB2 NODE : 000
APPID : G92B7027.H907.012C87185802
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrRetrySetup, probe:10
MESSAGE : Client Reroute is starting....
DATA #1 : Hexdump, 136 bytes
0x0012FCD8 : 5351 4C43 4120 2020 8800 0000 7F8A FFFF SQLCA
0x0012FCE8 : 2500 2AFF 2AFF 30FF 5443 502F 4950 FF53 %.**.0.TCP/IP.S
0x0012FCF8 : 4F43 4B45 5453 FF39 2E34 332E 3836 2E35 OCKETS.9.43.86.5
0x0012FD08 : 38FF 7265 6376 FF20 2020 2020 2020 2020 8.recv.
0x0012FD18 : 2020 2020 2020 2020 2020 2020 2020 2020
0x0012FD28 : 2020 2020 2020 2020 5351 4C4A 434D 4E20 SQLJCMN
0x0012FD38 : 1200 3681 1200 0000 0000 0000 0000 0000 ..6.....
0x0012FD48 : 0000 0000 0000 0000 2020 2020 2020 2020
0x0012FD58 : 2020 2020 2020 2020

2006-03-07-13.07.25.463000-480 I381150H1558 LEVEL: Warning
PID : 2416 TID : 2392 PROC : db2bp.exe
INSTANCE: DB2 NODE : 000
APPID : G92B7027.H907.012C87185802
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20
MESSAGE : Reconnecting to Hostname/IP Address -->
DATA #1 : Hexdump, 256 bytes
0x01584C7E : 4465 6E6D 6172 6B2E 6974 736F 736A 2E73 Denmark.itsosj.s
0x01584C8E : 616E 6A6F 7365 2E69 626D 2E63 6F6D 0000 anjose.ibm.com..
0x01584C9E : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584CAE : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584CBE : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584CCE : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584CDE : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584CEE : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584CFE : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584D0E : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584D1E : 0000 0000 0000 0000 0000 0000 0000 0000
0x01584D2E : 0000 0000 0000 0000 0000 0000 0000 0000

```

0x01584D3E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D4E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D5E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D6E : 0000 0000 0000 0000 0000 0000 0000 0000 .....

2006-03-07-13.07.25.473000-480 I382710H451          LEVEL: Warning
PID      : 2416                      TID : 2392        PROC : db2bp.exe
INSTANCE: DB2                      NODE : 000
APPID    : G92B7027.H907.012C87185802
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20
MESSAGE : Reconnecting to Service name/Port number -->
DATA #1 : Hexdump, 15 bytes
0x01584D7E : 3530 3030 3000 0000 0000 0000 0000 00    50000.....

2006-03-07-13.07.25.713000-480 I383163H1558          LEVEL: Warning
PID      : 2416                      TID : 2392        PROC : db2bp.exe
INSTANCE: DB2                      NODE : 000
APPID    : G92B7027.M807.012C87210725
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20
MESSAGE : Reconnecting to Hostname/IP Address -->
DATA #1 : Hexdump, 256 bytes
0x01584C7E : 6A61 6D61 6963 6100 0000 0000 0000 0000    jamaica.....
0x01584C8E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584C9E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584CAE : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584CBE : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584CCE : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584CDE : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584CEE : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584CFE : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D0E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D1E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D2E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D3E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D4E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D5E : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x01584D6E : 0000 0000 0000 0000 0000 0000 0000 0000 .....

2006-03-07-13.07.25.723000-480 I384723H451          LEVEL: Warning
PID      : 2416                      TID : 2392        PROC : db2bp.exe
INSTANCE: DB2                      NODE : 000
APPID    : G92B7027.M807.012C87210725
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20
MESSAGE : Reconnecting to Service name/Port number -->
DATA #1 : Hexdump, 15 bytes
0x01584D7E : 3530 3030 3000 0000 0000 0000 0000 00    50000.....

2006-03-07-13.07.25.803000-480 I385176H346          LEVEL: Warning
PID      : 2416                      TID : 2392        PROC : db2bp.exe

```

```
INSTANCE: DB2                      NODE : 000
APPID   : G92B7027.M907.012C87210727
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrcClientReroute, probe:998
MESSAGE : Client Reroute is completed successfully.
```

Example 4-80 get connection state command from the client machine

DB2 GET CONNECTION STATE

```
Database Connection State
Connection state      = Connectable and Connected
Connection mode      = SHARE
Local database alias  = SOURCE
Database name        = SOURCE
Hostname             = jamaica
Service name         = 50000
```

Example 4-81 LIST DB DIRECTORY command from the client machine

DB2 LIST DB DIRECTORY

```
System Database Directory

Number of entries in the directory = 4

Database 1 entry:
Database alias      = SOURCE
Database name       = SOURCE
Node name          = DENMARK
Database release level = a.00
Comment            =
Directory entry type = Remote
Catalog database partition number = -1
Alternate server hostname = denmark
Alternate server port number = 50000
.....
```

4.6.9 STEP TESTPTKO9: Verify Q replication up and running

In this step, verify that the unidirectional Q replication environment is operating normally by checking the Q Capture process on JAMAICA, the subscription states on the primary server JAMAICA and the secondary servers CLYDE and BONNIE.

Example 4-82 on page 179 shows the output of the **asnlccmd** command on the primary server JAMAICA indicating that Q Capture is functioning normally.
Example 4-83 on page 179 shows the four subscriptions (two each for each

secondary servers CLYDE and BONNIE) on the primary server JAMAICA having a STATE of active “A”.

Example 4-84 on page 180 and Example 4-85 on page 180 show the **asnlccmd** command and active subscription states on the secondary server CLYDE, while Example 4-86 on page 181 and Example 4-87 on page 181 correspond to the secondary server BONNIE.

Example 4-82 asnlccmd on the new primary server JAMAICA

asnlccmd capture_server=SOURCE capture_schema=SOURCE status

```
2006-03-07-13.22.54.742395 ASN0520I "AsnQCcmd" : "SOURCE" : "Initial" : The
STATUS command response: "HoldLThread" thread is in the "is waiting" state.
2006-03-07-13.22.54.742500 ASN0520I "AsnQCcmd" : "SOURCE" : "Initial" : The
STATUS command response: "AdminThread" thread is in the "is resting" state.
2006-03-07-13.22.54.742523 ASN0520I "AsnQCcmd" : "SOURCE" : "Initial" : The
STATUS command response: "PruneThread" thread is in the "is resting" state.
2006-03-07-13.22.54.742546 ASN0520I "AsnQCcmd" : "SOURCE" : "Initial" : The
STATUS command response: "WorkerThread" thread is in the "is doing work" state.
```

Example 4-83 Subscription states on the new primary server JAMAICA

CONNECT TO SOURCE USER db2inst9 using

Database Connection Information

```
Database server      = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = SOURCE
```

**SELECT SUBSTR(pubqmapname,1,15) AS pubqmapname, state FROM
SOURCE.IBMQREP_SENDQUEUES**

| PUBQMAPNAME | STATE |
|-------------|-------|
| S_TO_TA_MAP | A |
| S_TO_TB_MAP | A |

2 record(s) selected.

**SELECT SUBSTR(subname,1,15) AS subname, sub_id, state, has_loadphase FROM
SOURCE.IBMQREP_SUBS**

| SUBNAME | SUB_ID | STATE | HAS_LOADPHASE |
|---------------|--------|-------|---------------|
| SUBA-PRODUCTS | 1 | A | I |

| | | | |
|---------------|---|---|---|
| SUBA-SALES | 2 | A | I |
| SUBB-PRODUCTS | 3 | A | I |
| SUBB-SALES | 4 | A | I |

4 record(s) selected.

CONNECT RESET

DB20000I The SQL command completed successfully.

Example 4-84 asnqacmd on the secondary server CLYDE

asnqacmd apply_server=TARGETA apply_schema=TARGETA status

2006-03-07-13.23.46.465418 ASN0520I "AsnQAcmd" : "TARGETA" : "Initial" : The STATUS command response: "HoldLThread" thread is in the "is waiting" state.
 2006-03-07-13.23.46.465594 ASN0520I "AsnQAcmd" : "TARGETA" : "Initial" : The STATUS command response: "AdminThread" thread is in the "is resting" state.
 2006-03-07-13.23.46.465629 ASN0520I "AsnQAcmd" : "TARGETA" : "Initial" : The STATUS command response: "MonitorThread" thread is in the "is resting" state.
 2006-03-07-13.23.46.465653 ASN0520I "AsnQAcmd" : "TARGETA" : "Initial" : The STATUS command response: "BR00000" thread is in the "is doing work" state.

Example 4-85 Subscription states on the secondary server CLYDE

CONNECT TO TARGETA USER db2inst9 using

Database Connection Information

Database server = DB2/AIX64 9.1.0
 SQL authorization ID = DB2INST9
 Local database alias = TARGETA

SELECT SUBSTR(REPQMAPNAME,1,15) AS pubqmapname, state FROM TARGETA.IBMQREP_RECVQUEUES

| PUBQMAPNAME | STATE |
|-------------|-------|
| ----- | ---- |
| S_TO_TA_MAP | A |

1 record(s) selected.

SELECT SUBSTR(subname,1,15) AS subname, sub_id, state, has_loadphase FROM TARGETA.IBMQREP_TARGETS

| SUBNAME | SUB_ID | STATE | HAS_LOADPHASE |
|---------------|--------|-------|---------------|
| ----- | ----- | ---- | ----- |
| SUBA-PRODUCTS | 1 | A | I |
| SUBA-SALES | 2 | A | I |

2 record(s) selected.

CONNECT RESET
DB20000I The SQL command completed successfully.

Example 4-86 asnqacmd on the secondary server BONNIE

asnqacmd apply_server=TARGETB apply_schema=TARGETB status

2006-03-07-13.23.06.690018 ASN0520I "AsnQAcmd" : "TARGETB" : "Initial" : The
STATUS command response: "HoldLThread" thread is in the "is waiting" state.
2006-03-07-13.23.06.690188 ASN0520I "AsnQAcmd" : "TARGETB" : "Initial" : The
STATUS command response: "AdminThread" thread is in the "is resting" state.
2006-03-07-13.23.06.690213 ASN0520I "AsnQAcmd" : "TARGETB" : "Initial" : The
STATUS command response: "MonitorThread" thread is in the "is resting" state.
**2006-03-07-13.23.06.690236 ASN0520I "AsnQAcmd" : "TARGETB" : "Initial" : The
STATUS command response: "BR00000" thread is in the "is doing work" state.**

Example 4-87 Subscription states on the secondary server BONNIE

CONNECT TO TARGETB USER db2inst9 using

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = TARGETB

**SELECT SUBSTR(REPQMAPNAME,1,15) AS pubqmapname, state FROM
TARGETB.IBMQREP_RECQUEUEES**

| PUBQMAPNAME | STATE |
|-------------|-------|
| ----- | ----- |
| S_TO_TB_MAP | A |

1 record(s) selected.

**SELECT SUBSTR(subname,1,15) AS subname, sub_id, state, has_loadphase FROM
TARGETB.IBMQREP_TARGETS**

| SUBNAME | SUB_ID | STATE | HAS_LOADPHASE |
|---------------|--------|-------|---------------|
| ----- | ----- | ----- | ----- |
| SUBB-PRODUCTS | 3 | A | I |
| SUBB-SALES | 4 | A | I |

2 record(s) selected.

CONNECT RESET
DB20000I The SQL command completed successfully.

4.6.10 STEP TESTPTKO10: Verify all workload changes applied to target tables

In this step, verify that all the changes that have been applied on the old primary server DENMARK and new primary server JAMAICA (after planned takeover) by the application workload have been successfully applied to the secondary servers CLYDE and BONNIE.

Example 4-88 show the SQL to connect to the new primary server JAMAICA and list the contents of the PRODUCTS and SALES table, while Example 4-89 on page 183 and Example 4-90 on page 184 show the SQL to view the contents of the corresponding tables on the secondary servers CLYDE and BONNIE respectively.

Attention: The content of the SOURCE database on the new primary server JAMAICA fully matches the corresponding contents on the secondary servers CLYDE and BONNIE — this indicates that HADR and unidirectional replication coexist synergistically and seamlessly in the planned takeover scenario.

Example 4-88 Contents of source tables on the primary server JAMAICA

CONNECT TO SOURCE

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = SOURCE

SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|--------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | TOOLS | 8.00 | 200 |
| 30 | WHATCHAMACALLITS | TOOLS | 10.00 | 100 |
| 40 | DOODADS | TOOLS | 3.00 | 100 |
| 60 | HEAVY HAMMER | TOOLS | 100.00 | 100 |

5 record(s) selected.

```
SELECT * FROM DB2INST9.SALES WHERE REGION_CODE = 'EAST' ORDER BY REGION_CODE,
PRODUCT_ID
```

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | REGION_CODE |
|------------|----------|---------------|--------------|-------------|
| 10 | 100 | 1000 | 03/01/2006 | EAST |
| 20 | 120 | 2000 | 03/01/2006 | EAST |
| 30 | 100 | 5000 | 03/02/2006 | EAST |
| 60 | 160 | 6000 | 03/03/2006 | EAST |

4 record(s) selected.

```
SELECT * FROM DB2INST9.SALES WHERE REGION_CODE = 'WEST' ORDER BY REGION_CODE,
PRODUCT_ID
```

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | REGION_CODE |
|------------|----------|---------------|--------------|-------------|
| 10 | 200 | 1000 | 03/02/2006 | WEST |
| 20 | 220 | 2000 | 03/02/2006 | WEST |
| 30 | 200 | 5000 | 03/01/2006 | WEST |
| 30 | 200 | 3000 | 03/03/2006 | WEST |
| 40 | 140 | 4000 | 03/02/2006 | WEST |
| 40 | 240 | 4000 | 03/01/2006 | WEST |

6 record(s) selected.

Example 4-89 Contents of the target tables on the secondary server CLYDE

CONNECT TO TARGETA

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = TARGETA

```
SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID
```

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|--------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | TOOLS | 8.00 | 200 |
| 30 | WHATCHAMACALLITS | TOOLS | 10.00 | 100 |
| 40 | DOODADS | TOOLS | 3.00 | 100 |
| 60 | HEAVY HAMMER | TOOLS | 100.00 | 100 |

5 record(s) selected.

SELECT * FROM DB2INST9.SALES_EAST ORDER BY PRODUCT_ID

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE |
|------------|----------|---------------|--------------|
| 10 | 100 | 1000 | 03/01/2006 |
| 20 | 120 | 2000 | 03/01/2006 |
| 30 | 100 | 5000 | 03/02/2006 |
| 60 | 160 | 6000 | 03/03/2006 |

4 record(s) selected.

Example 4-90 Contents of the target tables on the secondary server BONNIE

CONNECT TO TARGETB

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = TARGETB

SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|--------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | TOOLS | 8.00 | 200 |
| 30 | WHATCHAMACALLITS | TOOLS | 10.00 | 100 |
| 40 | DOODADS | TOOLS | 3.00 | 100 |
| 60 | HEAVY HAMMER | TOOLS | 100.00 | 100 |

5 record(s) selected.

SELECT * FROM DB2INST9.SALES_WEST ORDER BY PRODUCT_ID

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE |
|------------|----------|---------------|--------------|
| 10 | 200 | 1000 | 03/02/2006 |
| 20 | 220 | 2000 | 03/02/2006 |
| 30 | 200 | 5000 | 03/01/2006 |
| 30 | 200 | 3000 | 03/03/2006 |
| 40 | 140 | 4000 | 03/02/2006 |
| 40 | 240 | 4000 | 03/01/2006 |

4.7 Unplanned takeover

The unplanned takeover 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 unplanned takeover.

The following subsections describe the recommended procedure for an unplanned takeover as shown in Figure 4-9 on page 186 and include steps to verify that the procedure ensures that unidirectional replication continues seamlessly without any data loss.

The assumptions made about this procedure are as follows:

- ▶ HADR is up and running with an application workload accessing the primary server DENMARK.
- ▶ DB2 Client Reroute is in effect and alternate server information is updated on the client machine.
- ▶ Application workload is not explicitly halted throughout the procedure — however, the client may experience some unavailability during the switchover of the primary to the standby server.
- ▶ Unidirectional replication has been up and running, but Q Capture is not started on the primary server DENMARK, and Q Apply is not started on the secondary servers CLYDE and BONNIE.

| |
|--|
| <u>STEP TESTUTKO1:</u> Start Q Apply on both secondary servers |
| <u>STEP TESTUTKO2:</u> Start Q Capture on HADR primary |
| <u>STEP TESTUTKO3:</u> Connect to HADR primary & view connection state |
| <u>STEP TESTUTKO4:</u> Run workload on client |
| <u>STEP TESTUTKO5:</u> Stop Q Capture on HADR primary & verify HADR status |
| <u>STEP TESTUTKO6:</u> Run workload on client to build unpropagated changes on HADR primary |
| <u>STEP TESTUTKO7:</u> Stop DB2 force on HADR primary server to simulate failure of HADR primary server |
| <u>STEP TESTUTKO8:</u> Takeover HADR on standby by FORCE & verify HADR status |
| <u>STEP TESTUTKO9:</u> Start Q Capture on HADR new primary (twice) |
| <u>STEP TESTUTKO10:</u> Run workload on client & verify connection state |
| <u>STEP TESTUTKO11:</u> Verify Q replication up & running |
| <u>STEP TESTUTKO12:</u> Verify all workload changes applied to target tables |

Figure 4-9 Overview of unplanned takeover steps

The steps described in Figure 4-9 are described in the following sections.

4.7.1 STEP TESTUTKO1: Start Q Apply on both secondary servers

In this step, start Q Apply on the secondary servers CLYDE and BONNIE as described in 4.6.1, “STEP TESTPTKO1: Start Q Apply on both secondary servers” on page 157.

4.7.2 STEP TESTUTKO2: Start Q Capture on HADR primary

In this step, start Q Capture on the HADR primary server DENMARK as described in 4.6.2, “STEP TESTPTKO2: Start Q Capture on HADR primary” on page 159.

4.7.3 STEP TESTUTKO3: Connect to HADR primary and view state

In this step, connect to the HADR primary server DENMARK from the client machine and view its connection state as described in 4.6.3, “STEP TESTPTKO3: Connect to HADR primary and view state” on page 160.

Note: As mentioned then, this step would typically not be part of a “real-world” planned takeover procedure — it is included here for you to better understand the state transitions that occur in a planned takeover scenario.

4.7.4 STEP TESTUTKO4: Run workload on client

In this step, run a workload on the client machine to simulate a “real-world” environment — this workload is identical to the one described in 4.6.4, “STEP TESTPTKO4: Run workload on client” on page 161.

4.7.5 STEP TESTUTKO5: Stop Q Capture on HADR primary and verify the HADR status

This step and the one described in 4.7.6, “STEP TESTUTKO6: Run workload on client to build unpropagated changes on HADR primary” on page 187 are meant to build messages in the DB2 log that have not yet been processed by the Q Capture program. This action creates conditions of unprocessed transactions on the DB2 log that exist when an unexpected failure occurs on the primary server.

The procedure described in 4.6.5, “STEP TESTPTKO5: Stop Q Capture on HADR primary and verify the HADR status” on page 162 can be used here.

4.7.6 STEP TESTUTKO6: Run workload on client to build unpropagated changes on HADR primary

In this step, run another workload on the client machine to build actually build the unprocessed transactions in the DB2 log.

Run the workload described in 4.6.8, “STEP TESTPTKO8: Run workload on client and verify state” on page 174.

4.7.7 STEP TESTUTKO7: Stop DB2 force on HADR primary to simulate failure of HADR primary

In this step, stop DB2 force to simulate an unplanned failure of the primary server DENMARK as shown in Example 4-91 on page 188.

Example 4-91 db2stop force command on the primary server DENMARK

db2stop force

```
03/07/2006 14:18:41    0    0    SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.
```

4.7.8 STEP TESTUTKO8: Takeover HADR on standby by FORCE and verify HADR status

After the simulated failure of the primary server DENMARK, the HADR status on the standby server JAMAICA is determined by issuing the **db2pd** command on the standby server JAMAICA as shown in Example 4-92. It shows the Role as still being “Standby” and the State as being “RemoteCatchupPending” from “Peer”.

Example 4-93 on page 189 shows the **TAKEOVER** command issued on the standby server JAMAICA to takeover the primary role from DENMARK by force. Example 4-94 on page 189 lists the contents of the db2diag.log on JAMAICA which documents the transition of the HADR Role from “Standby” and State “Peer” to the Role of “Primary” and State “Disconnected” (as shown in Example 4-95 on page 192) via “RemoteCatchupPending” (as shown in Example 4-92).

Example 4-92 Check HADR status on the standby server JAMAICA

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 3 days 22:25:39

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|----------------------|----------|------------------|----------------------|
| Standby | RemoteCatchupPending | Sync | 0 | 150 |

| ConnectStatus | ConnectTime | Timeout |
|---------------|--------------------------------------|---------|
| Disconnected | Tue Mar 7 14:18:41 2006 (1141762721) | 120 |

LocalHost
jamaica

LocalService
60001

| | | |
|-------------------|----------------------|-----------------------|
| RemoteHost | RemoteService | RemoteInstance |
| denmark | 60000 | db2inst9 |

PrimaryFile PrimaryPg PrimaryLSN
S0000071.LOG 605 0x000000001236DE86

StandByFile StandByPg StandByLSN
S0000071.LOG 605 0x000000001236DE86

Example 4-93 Issue Takeover HADR on database source by force

db2 TAKEOVER HADR ON DATABASE SOURCE BY FORCE;

DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.

Example 4-94 Contents of the db2diag.log on the standby server JAMAICA

2006-03-07-14.18.41.412608-360 I8694A333 LEVEL: Error
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrTcpIpRecv,
probe:11810
MESSAGE : Zero bytes received. Remote end may have closed connection

2006-03-07-14.18.41.412844-360 I9028A394 LEVEL: Error
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrRecvMsgS,
probe:30080
MESSAGE : HADR standby rcv error:
DATA #1 : Hexdump, 4 bytes
0x0FFFFFFFFFB5E0 : 0000 0001

2006-03-07-14.18.41.412993-360 I9423A380 LEVEL: Severe
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent,
probe:20215
RETCODE : ZRC=0x8280001B=-2105540581=HDR_ZRC_COMM_CLOSED
"Communication with HADR partner was lost"

2006-03-07-14.18.41.413138-360 E9804A330 LEVEL: Event
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to S-RemoteCatchupPending (was S-Peer)

2006-03-07-14.20.42.123434-360 I10135A489 LEVEL: Error
 PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent,
 probe:20200
 MESSAGE : Did not receive anything through HADR connection for the duration of
 HADR_TIMEOUT. Closing connection.
 DATA #1 : Hexdump, 4 bytes
 0x0FFFFFFFFFB718 : 0000 0079 ...y

2006-03-07-14.20.42.123717-360 I10625A314 LEVEL: Severe
 PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent,
 probe:20200
 RETCODE : ZRC=0x00000000=0=PSM_OK "Unknown"

2006-03-07-14.22.37.467393-360 I10940A396 LEVEL: Warning
 PID : 524506 TID : 1 PROC : db2agent (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-370 APPID: *LOCAL.db2inst9.060307202237
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
 probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-03-07-14.22.37.533416-360 I11337A327 LEVEL: Warning
 PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
 probe:47005
MESSAGE : Info: Standby has initiated a takeover by force.

2006-03-07-14.22.37.633608-360 I11665A320 LEVEL: Warning
 PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
 probe:47006
MESSAGE : Info: Standby switching roles to primary.

2006-03-07-14.22.37.636287-360 I11986A393 LEVEL: Info
 PID : 495856 TID : 1 PROC : db2redom (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-335
 FUNCTION: DB2 UDB, recovery manager, sqlpPRecReadLog, probe:1270
 DATA #1 : <preformatted>
 Moving nextLsn from 000000001236DE87 to scanNextMetaOut.shrNextLsn
 00000000124F800C

2006-03-07-14.22.37.644131-360 I12380A357 LEVEL: Info

PID : 450626 TID : 1 PROC : db2loggr (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, recovery manager, sqlpgSwitchFromRedoToUndo, probe:3220
 DATA #1 : <preformatted>
 nextLsn 00000000124F800C at the start of log page, extent header state is 1

2006-03-07-14.22.37.644389-360 I12738A396 LEVEL: Info
 PID : 495856 TID : 1 PROC : db2redom (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-335
 FUNCTION: DB2 UDB, recovery manager, sqlpPRecReadLog, probe:4630
 DATA #1 : <preformatted>
 nextLsn 00000000124F800C rfwd_ReadLsn 000000001236DE86 ExtNum 73 firstLsn 0000128E0000

2006-03-07-14.22.37.645496-360 I13135A341 LEVEL: Warning
 PID : 905342 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-335
 FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:1990
 DATA #1 : <preformatted>
 nextLsn 00000000124F800C

2006-03-07-14.22.37.753937-360 E13477A388 LEVEL: Warning
 PID : 905342 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-335
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:3600
MESSAGE : ADM1605I DB2 is invoking the backward phase of database rollforward recovery.

2006-03-07-14.22.37.754272-360 I13866A391 LEVEL: Warning
 PID : 905342 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-335
 FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:2210
 DATA #1 : <preformatted>
Invoking database rollforward backward recovery, nextLsn: 00000000124F800C

2006-03-07-14.22.37.898326-360 E14258A359 LEVEL: Warning
 PID : 905342 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-335
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:6600
MESSAGE : ADM1611W The rollforward recovery phase has been completed.

2006-03-07-14.22.37.898594-360 I14618A333 LEVEL: Warning
 PID : 905342 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE

```

APPHDL : 0-335
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:9500
MESSAGE : Stopping Replay Master on standby.

2006-03-07-14.22.38.107639-360 E14952A346          LEVEL: Event
PID      : 1069206          TID : 1          PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9          NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to P-RemoteCatchupPending (was S-RemoteCatchupPending)

2006-03-07-14.22.38.209510-360 I15299A330          LEVEL: Warning
PID      : 1069206          TID : 1          PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9          NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
probe:47007
MESSAGE : Info: Standby has completed takeover (now primary).

2006-03-07-14.22.38.210232-360 I15630A295          LEVEL: Warning
PID      : 1069206          TID : 1          PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9          NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20342
MESSAGE : Info: Standby Finished.

2006-03-07-14.22.38.311401-360 I15926A294          LEVEL: Warning
PID      : 1069206          TID : 1          PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9          NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20301
MESSAGE : Info: Primary Started.

```

Example 4-95 Check HADR status on the standby server JAMAICA

```
db2pd -db source -hadr
```

```

Database Partition 0 -- Database SOURCE -- Active -- Up 3 days 22:31:02

HADR Information:
Role      State          SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Primary   Disconnected        Sync      0                  0

ConnectStatus ConnectTime          Timeout
Disconnected  Tue Mar 7 14:22:37 2006 (1141762957) 120

LocalHost          LocalService
jamaica            60001

RemoteHost          RemoteService      RemoteInstance
denmark             60000             db2inst9

```

| | | |
|--------------|-----------|--------------------|
| PrimaryFile | PrimaryPg | PrimaryLSN |
| S0000072.LOG | 0 | 0x00000000124F8000 |
| StandByFile | StandByPg | StandByLSN |
| S0000000.LOG | 0 | 0x0000000000000000 |

4.7.9 STEP TESTUTKO9: Start the Q Capture on the HADR new primary (twice)

In this step, start Q Capture on the HADR new primary server JAMAICA as described in 4.6.7, “STEP TESTPTKO7: Start the Q Capture on the new HADR primary (twice)” on page 171.

4.7.10 STEP TESTUTKO10: Run workload on client and verify state

In this step, run another workload on the client machine as shown in Example 4-96, the output of which is shown in Example 4-97 on page 194. Next review the connection state and the database directory contents on the client machine as shown in Example 4-97 on page 194.

Note: In a “real-world” environment, the application workload is probably running continuously from a number of client machines.

As in the case of Example 4-78 on page 175, the output of the workload shown in Example 4-97 on page 194 shows that the first SQL statement “UPDATE DB2INST9.PRODUCTS SET PRODUCT_CATEGORY = ‘WEAPONS’ WHERE PRODUCT_ID = 80” encounters a problem connecting to the old primary server DENMARK, but DB2 Client Reroute automatically reroutes the query to the alternate server JAMAICA and succeeds. Message SQL301018N in Example 4-97 on page 194 indicates this occurrence. The contents of the db2diag.log on the client machine is similar to that shown in Example 4-79 on page 176 indicating the successful initialization and completion of DB2 Client Reroute. The output of the **get connection state** command issued from the client machine is identical to that shown in Example 4-80 on page 178 indicating that the client is now connected to the new primary server JAMAICA. The output of the **list db directory** command issued from the client machine is identical to that shown in Example 4-81 on page 178 indicating that the Alternate server hostname has been changed to DENMARK from JAMAICA because the roles have been switched.

Example 4-96 Run a workload on the client 1/2

```
db2 +c -tvf w3.sql
-- contents of w3.sql
UPDATE DB2INST9.PRODUCTS
SET PRODUCT_CATEGORY = 'WEAPONS' WHERE PRODUCT_ID = 80;

INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (80, 'FLAMETHROWERS', 'TOOLS', 1000.00, 300);

INSERT INTO DB2INST9.SALES
(PRODUCT_ID, STORE_ID, QUANTITY_SOLD, DATE_OF_SALE, REGION_CODE)
VALUES (80, 270, 5, '2006-03-04', 'WEST');

UPDATE DB2INST9.PRODUCTS
SET PRODUCT_CATEGORY = 'WEAPONS' WHERE PRODUCT_ID = 80;

UPDATE DB2INST9.SALES SET QUANTITY_SOLD = 6000
WHERE PRODUCT_ID =40 AND STORE_ID = 240 AND DATE_OF_SALE = '2006-03-01';

COMMIT;
```

Example 4-97 Run a workload on the client 2/2

```
UPDATE DB2INST9.PRODUCTS SET PRODUCT_CATEGORY = 'WEAPONS' WHERE PRODUCT_ID = 80
```

DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:
SQL30108N A connection failed but has been re-established. The hostname or IP address is "jamaica" and the service name or port number is "50000". Special registers may or may not be re-attempted (Reason code = "1"). SQLSTATE=08506

```
INSERT INTO DB2INST9.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 DB2INST9.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.

```
UPDATE DB2INST9.PRODUCTS SET PRODUCT_CATEGORY = 'WEAPONS' WHERE PRODUCT_ID = 80
```

DB20000I The SQL command completed successfully.

```
UPDATE DB2INST9.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.

COMMIT

DB20000I The SQL command completed successfully.

4.7.11 STEP TESTUTKO11: Verify Q replication up and running

In this step, verify that the unidirectional Q replication environment is operating normally by checking the Q Capture process on JAMAICA, the subscription states on the primary server JAMAICA and the secondary servers CLYDE and BONNIE as described in 4.6.9, “STEP TESTPTKO9: Verify Q replication up and running” on page 178.

4.7.12 STEP TESTUTKO12: Verify all workload changes applied to target tables

In this step, verify that all the changes that have been applied on the old primary server DENMARK and new primary server JAMAICA (after planned takeover) by the application workload have been successfully applied to the secondary servers CLYDE and BONNIE.

Example 4-98 show the SQL to connect to the new primary server JAMAICA and list the contents of the PRODUCTS and SALES table, while Example 4-99 on page 196 and Example 4-100 on page 197 show the SQL to view the contents of the corresponding tables on the secondary servers CLYDE and BONNIE respectively.

Attention: The content of the SOURCE database on the new primary server JAMAICA fully matches the corresponding contents on the secondary servers CLYDE and BONNIE — this indicates that HADR and unidirectional replication coexist synergistically and seamlessly in the unplanned takeover scenario.

Example 4-98 Contents of source tables on the primary server JAMAICA

CONNECT TO SOURCE

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = SOURCE

SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|---------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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 |

6 record(s) selected.

SELECT * FROM DB2INST9.SALES WHERE REGION_CODE = 'EAST' ORDER BY REGION_CODE, PRODUCT_ID

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | REGION_CODE |
|------------|----------|---------------|--------------|-------------|
| 10 | 100 | 1000 | 03/01/2006 | EAST |
| 20 | 120 | 2000 | 03/01/2006 | EAST |
| 30 | 100 | 5000 | 03/02/2006 | EAST |
| 60 | 160 | 6000 | 03/03/2006 | EAST |

4 record(s) selected.

SELECT * FROM DB2INST9.SALES WHERE REGION_CODE = 'WEST' ORDER BY REGION_CODE, PRODUCT_ID

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | REGION_CODE |
|------------|----------|---------------|--------------|-------------|
| 10 | 200 | 1000 | 03/02/2006 | WEST |
| 20 | 220 | 2000 | 03/02/2006 | WEST |
| 30 | 200 | 5000 | 03/01/2006 | WEST |
| 30 | 200 | 3000 | 03/03/2006 | WEST |
| 40 | 140 | 4000 | 03/02/2006 | WEST |
| 40 | 240 | 6000 | 03/01/2006 | WEST |
| 80 | 270 | 5 | 03/04/2006 | WEST |

7 record(s) selected.

Example 4-99 Contents of the target tables on the secondary server CLYDE

CONNECT TO TARGETA

Database Connection Information

Database server = DB2/AIX64 9.1.0

```

SQL authorization ID  = DB2INST9
Local database alias  = TARGETA

```

```
SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID
```

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|---------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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 |

6 record(s) selected.

```
SELECT * FROM DB2INST9.SALES_EAST ORDER BY PRODUCT_ID
```

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE |
|------------|----------|---------------|--------------|
| 10 | 100 | 1000 | 03/01/2006 |
| 20 | 120 | 2000 | 03/01/2006 |
| 30 | 100 | 5000 | 03/02/2006 |
| 60 | 160 | 6000 | 03/03/2006 |

4 record(s) selected.

Example 4-100 Contents of the target tables on the secondary server BONNIE

```
CONNECT TO TARGETB
```

Database Connection Information

```

Database server      = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = TARGETB

```

```
SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID
```

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|---------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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 |

6 record(s) selected.

```
SELECT * FROM DB2INST9.SALES_WEST ORDER BY PRODUCT_ID
```

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | |
|------------|----------|---------------|--------------|------|
| 10 | 200 | 1000 | 03/02/2006 | WEST |
| 20 | 220 | 2000 | 03/02/2006 | WEST |
| 30 | 200 | 5000 | 03/01/2006 | WEST |
| 30 | 200 | 3000 | 03/03/2006 | WEST |
| 40 | 140 | 4000 | 03/02/2006 | WEST |
| 40 | 240 | 6000 | 03/01/2006 | WEST |
| 80 | 270 | 5 | 03/04/2006 | WEST |

7 record(s) selected.

4.8 Failback

This section describes the failback procedure for restoring the HADR and unidirectional Q replication environment to its pre-planned/unplanned takeover normal operating environment, where the primary server is DENMARK and the standby server is JAMAICA. Here again, this process should involve no loss of any data.

The following subsections describe the recommended procedure for failback as shown in Figure 4-10 on page 199 and include steps to verify that the procedure ensures that unidirectional replication continues seamlessly without any data loss.

The assumptions made about this procedure are as follows:

- ▶ DB2 database on DENMARK is intact, otherwise it would be similar to a setting up the HADR environment afresh, where the database on JAMAICA would need to be backed up and restored on DENMARK as described in “Set up HADR between the primary and standby server” on page 122.
- ▶ HADR is up and running with an application workload accessing the primary server JAMAICA.
- ▶ DB2 Client Reroute is in effect and alternate server information (referencing DENMARK) is updated on the client machine.
- ▶ Application workload is not explicitly halted throughout the procedure — however, the client may experience some unavailability during the switchover of the primary to the standby server roles.

- Unidirectional replication has been up and running.

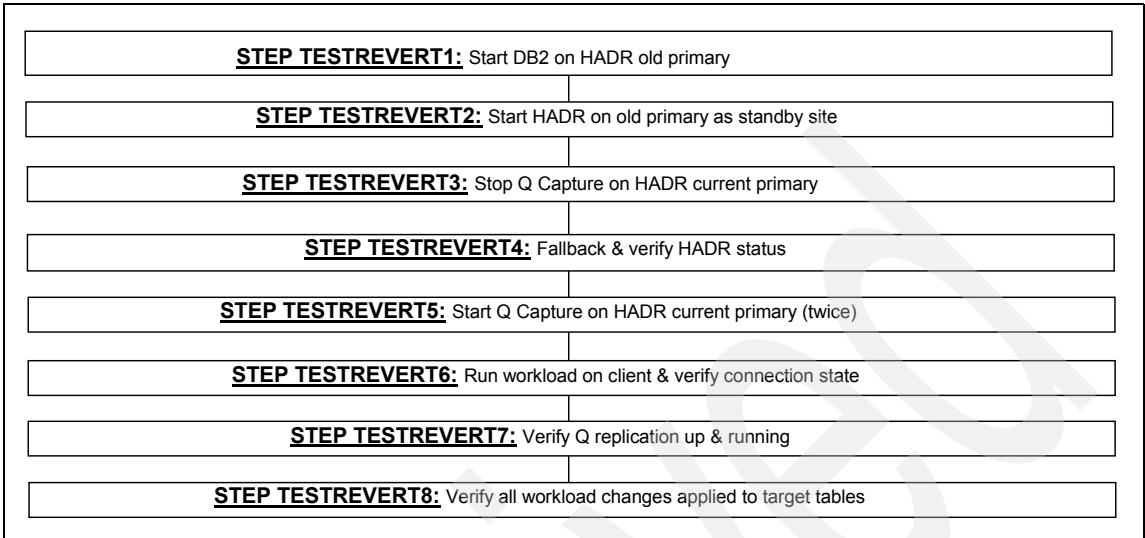


Figure 4-10 Overview of fallback

The steps described in Figure 4-10 are described in the following sections.

4.8.1 STEP TESTREVERT1: Start DB2 on HADR old primary

Start DB2 on the old primary server DENMARK as shown in Example 4-101.

Example 4-101 Start DB2 on the old primary server DENMARK

db2start

```
03/07/2006 17:39:13    0    0    SQL1063N  DB2START processing was successful.  
SQL1063N  DB2START processing was successful.
```

4.8.2 STEP TESTREVERT2: Start HADR on old primary as standby site and verify HADR status

In this step, issue the TAKEOVER command on the old primary server DENMARK to be the standby server to the primary server JAMAICA by executing the command shown in Example 4-102 on page 200. The db2diag.log files on the old primary server DENMARK (Example 4-103 on page 200) and the primary server JAMAICA (Example 4-104 on page 204) document in detail the progress of the takeover process, where the old primary server DENMARK again

becomes the standby server, and the current primary server JAMAICA remains as the primary server.

Example 4-105 on page 205 and Example 4-106 on page 206 show the HADR status of the old primary server DENMARK and the current primary server JAMAICA after takeover processing has completed. The **db2pd** command issued on the now standby server DENMARK (Example 4-105 on page 205) shows the HADR Role to be “Standby”, State to be “Peer” and the SyncMode to be “Sync”. The **db2pd** command issued on the current primary server JAMAICA (Example 4-106 on page 206) shows the HADR Role to be “Primary”, State to be “Peer” and the SyncMode to be “Sync”.

Example 4-102 Start HADR on database source as standby command on DENMARK

db2 START HADR ON DATABASE SOURCE AS STANDBY

DB20000I The START HADR ON DATABASE command completed successfully.

Example 4-103 Contents of the db2diag.log on the DENMARK

.....

2006-03-07-17.41.14.810062-360 I5155A374 LEVEL: Warning
PID : 1237042 TID : 1 PROC : db2agent (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
APPHDL : 0-7 APPID: *LOCAL.db2inst9.060307234114
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-03-07-17.41.14.837211-360 E5530A310 LEVEL: Event
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to None (was None)

2006-03-07-17.41.14.845442-360 I5841A332 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrResolveHostNamesToIp, probe:20100
MESSAGE : HADR_LOCAL_HOST denmark mapped to 9.43.86.58

2006-03-07-17.41.14.845917-360 I6174A333 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000

FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrResolveHostNamesToIp, probe:20110
MESSAGE : HADR_REMOTE_HOST jamaica mapped to 9.43.86.55

2006-03-07-17.41.14.846056-360 I6508A299 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHosts,
probe:20157
MESSAGE : HADR is using IPv4.

2006-03-07-17.41.14.846467-360 E6808A312 LEVEL: Event
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to S-Boot (was None)

2006-03-07-17.41.14.846606-360 I7121A315 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster,
probe:21251
MESSAGE : Info: Replaymaster Starting...

2006-03-07-17.41.14.850171-360 I7437A321 LEVEL: Warning
PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
APPHDL : 0-8
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-03-07-17.41.14.851662-360 I7759A330 LEVEL: Warning
PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:300
MESSAGE : Starting Replay Master on standby.

2006-03-07-17.41.14.851895-360 I8090A317 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster,
probe:21252
MESSAGE : Info: Replaymaster request done.

2006-03-07-17.41.14.852044-360 I8408A294 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0

INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20341
MESSAGE : Info: Standby Started.

2006-03-07-17.41.14.853008-360 I8703A406 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrHandleHsAck,
probe:30445
**MESSAGE : HADR: old primary reintegration as new standby discarding obsolete
logs after hdrLCUEndLsnRequested 00000000124F7FFB**

2006-03-07-17.41.14.853179-360 E9110A322 LEVEL: Event
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to S-LocalCatchup (was S-Boot)

2006-03-07-17.41.14.854656-360 E9433A346 LEVEL: Warning
PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:920
MESSAGE : ADM1602W Rollforward recovery has been initiated.

2006-03-07-17.41.14.854915-360 E9780A389 LEVEL: Warning
PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:1740
**MESSAGE : ADM1603I DB2 is invoking the forward phase of the database
rollforward recovery.**

2006-03-07-17.41.14.855144-360 I10170A417 LEVEL: Warning
PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:710
DATA #1 : <preformatted>
Invoking database rollforward forward recovery,
lowtranlsn 00000000124F800C minbufflsn 00000000124F800C

2006-03-07-17.41.14.874264-360 I10588A372 LEVEL: Warning
PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
DATA #1 : <preformatted>

Using parallel recovery with 5 agents 7 QSets 28 queues and 2 chunks

2006-03-07-17.41.14.888383-360 I10961A338 LEVEL: Warning
PID : 1085618 TID : 1 PROC : db2redom (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlpParallelRecovery, probe:510
MESSAGE : hdrLCUEndLsnRequested 00000000124F7FFB

2006-03-07-17.41.14.889030-360 E11300A368 LEVEL: Warning
PID : 1085618 TID : 1 PROC : db2redom (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-8
FUNCTION: DB2 UDB, recovery manager, sqlpParallelRecovery, probe:880
DATA #1 : <preformatted>
Resetting max shredder memory to 2195322 from 2867200

2006-03-07-17.41.14.953100-360 E11669A338 LEVEL: Event
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to S-RemoteCatchupPending (was S-LocalCatchup)

2006-03-07-17.41.15.082833-360 I12008A369 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20895
**MESSAGE : Pair validation passed. Primary reintegration: hdrLCUEndLsnRequested:
00000000124F7FFB**

2006-03-07-17.41.15.082999-360 I12378A361 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrNukeLogTail,
probe:10445
**MESSAGE : Primary reintegration: hdrNukeLogTail() called at LSN:
00000000124F7FFB**

2006-03-07-17.41.15.083829-360 I12740A364 LEVEL: Severe
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrNukeLogTail,
probe:10500
**MESSAGE : Error: reintegration could not read takeover LSN log page
00000000124F8000.**

2006-03-07-17.41.15.136585-360 E13105A339 LEVEL: Event
PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0

INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
 probe:10000
CHANGE : HADR state set to S-RemoteCatchup (was S-RemoteCatchupPending)

2006-03-07-17.41.15.136728-360 I13445A312 LEVEL: Warning
 PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSPrepareLogWrite,
 probe:10260
MESSAGE : RCUStartLsn 00000000124F800C

2006-03-07-17.41.14.852050-360 I13758A378 LEVEL: Warning
 PID : 1237042 TID : 1 PROC : db2agent (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 APPHDL : 0-7 APPID: *LOCAL.db2inst9.060307234114
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
 probe:21152
MESSAGE : Info: HADR Startup has completed.

2006-03-07-17.41.17.020066-360 E14137A329 LEVEL: Event
 PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
 probe:10000
CHANGE : HADR state set to S-NearlyPeer (was S-RemoteCatchup)

2006-03-07-17.41.17.110412-360 E14467A320 LEVEL: Event
 PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
 probe:10000
CHANGE : HADR state set to S-Peer (was S-NearlyPeer)

Example 4-104 Contents of the db2diag.log on JAMAICA

.....
 2006-03-07-17.41.15.052793-360 I1102A327 LEVEL: Warning
 PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20482
MESSAGE : 01d primary requesting rejoining HADR pair as a standby

2006-03-07-17.41.16.771625-360 E1430A339 LEVEL: Event
 PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
 probe:10000

CHANGE : HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)

2006-03-07-17.41.16.774247-360 I1770A313 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20445
MESSAGE : remote catchup starts at 00000000124F800C

2006-03-07-17.41.17.007885-360 I2084A330 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrTransitionPtoNPeer,
probe:10645
MESSAGE : near peer catchup starts at 0000000012531124

2006-03-07-17.41.17.108074-360 E2415A329 LEVEL: Event
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to P-NearlyPeer (was P-RemoteCatchup)

2006-03-07-17.41.17.109890-360 E2745A320 LEVEL: Event
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to P-Peer (was P-NearlyPeer)

Example 4-105 Check HADR status on DENMARK

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 0 days 00:24:53

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Standby | Peer | Sync | 0 | 46 |

| ConnectStatus | ConnectTime | Timeout |
|---------------|--------------------------------------|---------|
| Connected | Tue Mar 7 17:41:14 2006 (1141774874) | 120 |

| LocalHost | LocalService |
|-----------|--------------|
| denmark | 60000 |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| jamaica | 60001 | db2inst9 |

| | | |
|--------------|-----------|--------------------|
| PrimaryFile | PrimaryPg | PrimaryLSN |
| S0000072.LOG | 128 | 0x00000000125789B0 |
| StandByFile | StandByPg | StandByLSN |
| S0000072.LOG | 128 | 0x00000000125789B0 |

Example 4-106 Check HADR status on JAMAICA

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 4 days 02:11:14

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer | Sync | 0 | 28 |

| ConnectStatus | ConnectTime | Timeout |
|---------------|--------------------------------------|---------|
| Connected | Tue Mar 7 17:41:14 2006 (1141774874) | 120 |

| LocalHost | LocalService |
|-----------|--------------|
| jamaica | 60001 |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| denmark | 60000 | db2inst9 |

| | | |
|--------------|-----------|--------------------|
| PrimaryFile | PrimaryPg | PrimaryLSN |
| S0000072.LOG | 128 | 0x00000000125789B0 |
| StandByFile | StandByPg | StandByLSN |
| S0000072.LOG | 128 | 0x00000000125789B0 |

4.8.3 STEP TESTREVERT3: Stop Q Capture on HADR current primary

In this step, stop Q Capture on the current primary server JAMAICA by executing the command shown in Example 4-107 — stdout of Q Capture is shown in Example 4-108. You can monitor the contents of the Q Capture log which shows the program as having stopped as shown in Example 4-109 on page 207.

Example 4-107 Stop Q Capture on JAMAICA

asnqccmd capture_server=SOURCE capture_schema=SOURCE stop

2006-03-07-18.08.18.087786 ASN0522I "AsnQCcmd" : "SOURCE" : "Initial" : The program received the "STOP" command.

Example 4-108 Stop Q Capture stdout on the current primary server JAMAICA

```
2006-03-07-17.06.52.494484 ASN7000I "Q Capture" : "SOURCE" : "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.
2006-03-07-17.06.52.499955 ASN0572I "Q Capture" : "SOURCE" : "WorkerThread" :
The program initialized successfully.
2006-03-07-18.08.20.216720 ASN0573I "Q Capture" : "SOURCE" : "Initial" : The
program was stopped.
[asnqwk] Leaving the asnqwk thread:
[asnqwk] total database transactions captured = 0
[asnqwk] total MQSeries transactions          = 0
[asnqwk] total DB transactions published      = 0
[asnqwk] total time WAITING for logrd        = 3685150 seconds
[asntxrd] Leaving logtxrd reader thread:
[asntxrd] 0 message(s) in notification queue
[asntxrd] transmgr memory in use              = 0 bytes
[asntxrd] totalLogReadTime                    = 0.658 second(s)
[asntxrd] total logrd SLEEP time              = 3685.007 second(s)
[asntxrd] totalLogRecsRead                    = 0
[asntxrd] Queued a thread Termination Notification.
[waitForLogrdEnd] The logrd thread terminated with rc: 0
```

Example 4-109 Stop Q Capture log contents on the current primary server JAMAICA

```
.....
"WorkerThread" : At program termination, the highest log sequence number of a
successfully processed transaction is "440E:10EA:0000:0001:0000" and the lowest
log sequence number of a transaction still to be committed is
"0000:0000:0000:1257:8BF1".
.....
2006-03-07-18.08.20.216720 <asnqcap::main> ASN0573I "Q Capture" : "SOURCE" :
"Initial" : The program was stopped.
```

4.8.4 STEP TESTREVERT4: Fallback and verify HADR status

In this step, issue the **TAKEOVER** command on the old primary server DENMARK to switch the primary role from the current primary server JAMAICA as shown in Example 4-110 on page 208. The db2diag.log files on the old primary server DENMARK (Example 4-111 on page 208) and the primary server JAMAICA (Example 4-112 on page 210) document in detail the progress of the takeover process, where the old primary server DENMARK becomes the primary server again, and the primary server JAMAICA becomes the standby server — a switch of roles, which re-establishes the original HADR pair configuration.

Example 4-113 on page 213 and Example 4-114 on page 213 show the HADR status of the primary server and standby server after takeover processing has completed. The **db2pd** command issued on the primary server DENMARK (Example 4-113 on page 213) shows the HADR Role to be “Primary”, State to be “Peer” and the SyncMode to be “Sync”. The **db2pd** command issued on the standby server JAMAICA (Example 4-114 on page 213) shows the HADR Role to be “Standby”, State to be “Peer” and the SyncMode to be “Sync”. This corresponds to a original HADR operating environment.

Example 4-110 Issue Takeover HADR on database source command on DENMARK

db2 TAKEOVER HADR ON DATABASE SOURCE

DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.

Example 4-111 Contents of the db2diag.log on DENMARK

2006-03-07-18.09.47.868362-360 I1169A374 LEVEL: Warning
 PID : 1237042 TID : 1 PROC : db2agent (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 APPHDL : 0-54 APPID: *LOCAL.db2inst9.060308000947
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
 probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-03-07-18.09.47.946368-360 I1544A318 LEVEL: Warning
 PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
 probe:47001
MESSAGE : Info: Standby has initiated a takeover.

2006-03-07-18.09.48.084016-360 I1863A320 LEVEL: Warning
 PID : 901186 TID : 1 PROC : db2hadrs (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover,
 probe:47002
MESSAGE : Info: Standby switching roles to primary.

2006-03-07-18.09.48.096954-360 I2184A394 LEVEL: Info
 PID : 1085618 TID : 1 PROC : db2redom (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-8
 FUNCTION: DB2 UDB, recovery manager, sqlpPRecReadLog, probe:4630
 DATA #1 : <preformatted>
 nextLsn 0000000012578E1A rfw_ReadLsn 0000000012578E19 ExtNum 73 firstLsn
 0000128E0000

2006-03-07-18.09.48.098448-360 I2579A339 LEVEL: Warning

PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-8
 FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:1990
 DATA #1 : <preformatted>
 nextLsn 0000000012578E1A

2006-03-07-18.09.48.340829-360 E2919A386 LEVEL: Warning
 PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-8
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:3600
MESSAGE : ADM1605I DB2 is invoking the backward phase of database rollforward recovery.

2006-03-07-18.09.48.341170-360 I3306A389 LEVEL: Warning
 PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-8
 FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:2210
 DATA #1 : <preformatted>
Invoking database rollforward backward recovery, nextLsn: 0000000012578E1A

2006-03-07-18.09.48.779048-360 E3696A357 LEVEL: Warning
 PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-8
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:6600
MESSAGE : ADM1611W The rollforward recovery phase has been completed.

2006-03-07-18.09.48.779316-360 I4054A331 LEVEL: Warning
 PID : 413932 TID : 1 PROC : db2agnti (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000 DB : SOURCE
 APPHDL : 0-8
 FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:9500
MESSAGE : Stopping Replay Master on standby.

2006-03-07-18.09.49.371669-360 E4386A314 LEVEL: Event
 PID : 901186 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to P-Peer (was S-Peer)

2006-03-07-18.09.49.809250-360 I4701A330 LEVEL: Warning
 PID : 901186 TID : 1 PROC : db2hadrp (SOURCE) 0
 INSTANCE: db2inst9 NODE : 000
 FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSDoTakeover, probe:47003

MESSAGE : Info: Standby has completed takeover (now primary).

2006-03-07-18.09.49.809948-360 I5032A295 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20342
MESSAGE : Info: Standby Finished.

2006-03-07-18.09.50.210820-360 I5328A294 LEVEL: Warning
PID : 901186 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20301
MESSAGE : Info: Primary Started.

Example 4-112 Contents of the db2diag.log on JAMAICA

2006-03-07-18.09.48.046097-360 I1102A333 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrPDotakeover,
probe:43006
MESSAGE : Info: Primary has started non-forced takeover request.

2006-03-07-18.09.48.064603-360 I1436A322 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrPDotakeover,
probe:43002
MESSAGE : Info: Primary is switching to standby role.

2006-03-07-18.09.48.065259-360 E1759A400 LEVEL: Event
PID : 524506 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-680
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSwitchDbFromRuntimeToStandby, probe:50122
CHANGE : Current log position at time of Role Switch - LSN = 0000000012578E1A

2006-03-07-18.09.48.083090-360 E2160A352 LEVEL: Event
PID : 524506 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-680
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE : HADR state set to S-Peer (was P-Peer)

2006-03-07-18.09.48.083321-360 I2513A330 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrp (SOURCE) 0
INSTANCE: db2inst9 NODE : 000

FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrPDOTakeover,
probe:43003

MESSAGE : Info: Primary has completed takeover (now standby).

2006-03-07-18.09.48.084247-360 I2844A315 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster,
probe:21251

MESSAGE : Info: Replaymaster Starting...

2006-03-07-18.09.48.084583-360 I3160A345 LEVEL: Warning
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21151

MESSAGE : Info: HADR Startup has begun.

2006-03-07-18.09.48.084784-360 I3506A306 LEVEL: Severe
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, base sys utilities, sqlesrsu, probe:999
MESSAGE : free tran stuff

2006-03-07-18.09.48.085097-360 I3813A332 LEVEL: Warning
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:300
MESSAGE : Starting Replay Master on standby.

2006-03-07-18.09.48.085283-360 I4146A317 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster,
probe:21252
MESSAGE : Info: Replaymaster request done.

2006-03-07-18.09.48.085403-360 I4464A295 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20302
MESSAGE : Info: Primary Finished.

2006-03-07-18.09.48.085504-360 I4760A294 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000

FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20341

MESSAGE : Info: Standby Started.

2006-03-07-18.09.48.085642-360 I5055A312 LEVEL: Warning
PID : 1069206 TID : 1 PROC : db2hadrs (SOURCE) 0
INSTANCE: db2inst9 NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSPrepareLogWrite,
probe:10260
MESSAGE : RCUStartLsn 0000000012578E1A

2006-03-07-18.09.49.810117-360 E5368A348 LEVEL: Warning
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:920
MESSAGE : ADM1602W Rollforward recovery has been initiated.

2006-03-07-18.09.49.810373-360 E5717A391 LEVEL: Warning
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:1740
**MESSAGE : ADM1603I DB2 is invoking the forward phase of the database
rollforward recovery.**

2006-03-07-18.09.49.810600-360 I6109A419 LEVEL: Warning
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:710
DATA #1 : <preformatted>
**Invoking database rollforward forward recovery,
lowtranlsn 0000000012578E1A minbufflsn 00000000124F800C**

2006-03-07-18.09.49.822207-360 I6529A374 LEVEL: Warning
PID : 925782 TID : 1 PROC : db2agnti (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
DATA #1 : <preformatted>
Using parallel recovery with 5 agents 7 QSets 28 queues and 2 chunks

2006-03-07-18.09.49.822649-360 E6904A370 LEVEL: Warning
PID : 1024034 TID : 1 PROC : db2redom (SOURCE) 0
INSTANCE: db2inst9 NODE : 000 DB : SOURCE
APPHDL : 0-684
FUNCTION: DB2 UDB, recovery manager, sqlpParallelRecovery, probe:880
DATA #1 : <preformatted>

Resetting max shredder memory to 2113564 from 2867200

Example 4-113 Check HADR status on DENMARK

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 0 days 00:29:42

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer | Sync | 0 | 0 |

| ConnectStatus | ConnectTime | Timeout |
|---------------|--------------------------------------|---------|
| Connected | Tue Mar 7 17:41:14 2006 (1141774874) | 120 |

| LocalHost | LocalService |
|-----------|--------------|
| denmark | 60000 |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| jamaica | 60001 | db2inst9 |

| PrimaryFile | PrimaryPg | PrimaryLSN |
|--------------|-----------|--------------------|
| S0000072.LOG | 128 | 0x0000000012578E19 |

| StandByFile | StandByPg | StandByLSN |
|--------------|-----------|--------------------|
| S0000072.LOG | 128 | 0x0000000012578E19 |

Example 4-114 Check HADR status on JAMAICA

db2pd -db source -hadr

Database Partition 0 -- Database SOURCE -- Active -- Up 4 days 02:16:16

HADR Information:

| Role | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Standby | Peer | Sync | 0 | 0 |

| ConnectStatus | ConnectTime | Timeout |
|---------------|--------------------------------------|---------|
| Connected | Tue Mar 7 17:41:14 2006 (1141774874) | 120 |

| LocalHost | LocalService |
|-----------|--------------|
| jamaica | 60001 |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| denmark | 60000 | db2inst9 |

| PrimaryFile | PrimaryPg | PrimaryLSN |
|-------------|-----------|------------|
| | | |

```
S0000072.LOG 128      0x0000000012578E19

StandByFile StandByPg StandByLSN
S0000072.LOG 128      0x0000000012578E19
```

4.8.5 STEP TESTREVERT5: Start Q Capture on HADR primary (twice)

In this step, start Q Capture on the primary server DENMARK twice similar to that described for JAMAICA in 4.6.7, “STEP TESTPTKO7: Start the Q Capture on the new HADR primary (twice)” on page 171.

4.8.6 STEP TESTREVERT6: Run workload on client and verify state

In this step, run another workload on the client machine as shown in Example 4-115, the output of which is shown in Example 4-116 on page 215. Next review the connection state and the database directory contents on the client machine as shown in Example 4-118 on page 217 and Example 4-119 on page 218.

Note: In a “real-world” environment, the application workload is probably running continuously from a number of client machines.

As in the case of Example 4-78 on page 175, the output of the workload shown in Example 4-116 on page 215 shows that the first SQL statement “UPDATE DB2INST9.SALES SET QUANTITY_SOLD = 10 WHERE PRODUCT_ID = 10 AND STORE_ID = 200 AND DATE_OF_SALE = '2006-03-02'” encounters a problem connecting to the previous primary server JAMAICA, but DB2 Client Reroute automatically reroutes the query to the alternate server DENMARK and succeeds. Message SQL301018N in Example 4-116 on page 215 indicates this occurrence. Example 4-117 on page 215 shows the content of the db2diag.log on the client machine indicating the successful initialization and completion of DB2 Client Reroute.

Example 4-118 on page 217 shows the **get connection state** command issued from the client machine indicating that the client is now connected to the primary server DENMARK.

Example 4-119 on page 218 shows the **LIST DB DIRECTORY** command issued from the client machine indicating that the Alternate server hostname has been changed to JAMAICA from DENMARK because the roles have been switched.

Example 4-115 Run a workload on the client 1/2

```
db2 +c -tvf w4.sql
```

```
--contents of w4.sql
UPDATE DB2INST9.SALES
SET QUANTITY_SOLD = 10
WHERE PRODUCT_ID = 10 AND STORE_ID = 200 AND DATE_OF_SALE = '2006-03-02';

INSERT INTO DB2INST9.PRODUCTS
(PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, MANUFACT_ID)
VALUES (90, 'STRAWMEN', 'TOOLS', 8.00, 100);

UPDATE DB2INST9.SALES
SET QUANTITY_SOLD = 0
WHERE PRODUCT_ID = 10 AND STORE_ID = 200 AND DATE_OF_SALE = '2006-03-02';

COMMIT;
```

Example 4-116 Run a workload on the client 2/2

```
UPDATE DB2INST9.SALES SET QUANTITY_SOLD = 10 WHERE PRODUCT_ID = 10 AND STORE_ID
= 200 AND DATE_OF_SALE = '2006-03-02'
```

DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:
SQL30108N A connection failed but has been re-established. The hostname or IP address is "denmark" and the service name or port number is "50000". Special registers may or may not be re-attempted (Reason code = "1"). SQLSTATE=08506

```
INSERT INTO DB2INST9.PRODUCTS (PRODUCT_ID, PRODUCT_NAME, PRODUCT_CATEGORY,
PRICE, MANUFACT_ID) VALUES (90, 'STRAWMEN', 'TOOLS', 8.00, 100)
```

DB20000I The SQL command completed successfully.

```
UPDATE DB2INST9.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.

COMMIT

DB20000I The SQL command completed successfully.

Example 4-117 db2diag.log contents on the client machine

```
2006-03-07-18.13.01.301000-480 I396452H1019      LEVEL: Warning
PID      : 2232                      TID   : 2252      PROC  : db2bp.exe
INSTANCE: DB2                      NODE  : 000
APPID    : G92B7027.HD04.011848015512
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrRetrySetup, probe:10
MESSAGE  : Client Reroute is starting....
```

```

DATA #1 : Hexdump, 136 bytes
0x0012FCD8 : 5351 4C43 4120 2020 8800 0000 7F8A FFFF   SQLCA .....
0x0012FCE8 : 2500 2AFF 2AFF 30FF 5443 502F 4950 FF53   %.*.*.O.TCP/IP.S
0x0012FCF8 : 4F43 4B45 5453 FF39 2E34 332E 3836 2E35   OCKETS.9.43.86.5
0x0012FD08 : 35FF 7265 6376 FF20 2020 2020 2020 2020   5.recv.
0x0012FD18 : 2020 2020 2020 2020 2020 2020 2020 2020
0x0012FD28 : 2020 2020 2020 2020 5351 4C4A 434D 4E20           SQLJCMN
0x0012FD38 : 1200 3681 1200 0000 0000 0000 0000 0000   ..6.....
0x0012FD48 : 0000 0000 0000 0000 2020 2020 2020 2020   .....
0x0012FD58 : 2020 2020 2020 2020

2006-03-07-18.13.01.401000-480 I397473H1558      LEVEL: Warning
PID      : 2232                      TID : 2252      PROC : db2bp.exe
INSTANCE: DB2                      NODE : 000
APPID    : G92B7027.HD04.011848015512
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrlstToConnect, probe:20
MESSAGE : Reconnecting to Hostname/IP Address -->
DATA #1 : Hexdump, 256 bytes
0x01584C7E : 4A61 6D61 6963 612E 6974 736F 736A 2E73   Jamaica.itsosj.s
0x01584C8E : 616E 6A6F 7365 2E69 626D 2E63 6F6D 0000   anjose.ibm.com..
0x01584C9E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CAE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CBE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CCE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CDE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CEE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584CFE : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D0E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D1E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D2E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D3E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D4E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D5E : 0000 0000 0000 0000 0000 0000 0000 0000   .....
0x01584D6E : 0000 0000 0000 0000 0000 0000 0000 0000   .....

2006-03-07-18.13.01.411000-480 I399033H451      LEVEL: Warning
PID      : 2232                      TID : 2252      PROC : db2bp.exe
INSTANCE: DB2                      NODE : 000
APPID    : G92B7027.HD04.011848015512
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrlstToConnect, probe:20
MESSAGE : Reconnecting to Service name/Port number -->
DATA #1 : Hexdump, 15 bytes
0x01584D7E : 3530 3030 3000 0000 0000 0000 0000 00      50000.....

2006-03-07-18.13.01.511000-480 I399486H1558      LEVEL: Warning
PID      : 2232                      TID : 2252      PROC : db2bp.exe
INSTANCE: DB2                      NODE : 000
APPID    : G92B7027.I304.011848021301
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrlstToConnect, probe:20

```

MESSAGE : Reconnecting to Hostname/IP Address -->

DATA #1 : Hexdump, 256 bytes

```
0x01584C7E : 6465 6E6D 6172 6800 0000 0000 0000 0000    denmark.....
0x01584C8E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584C9E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584CAE : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584CBE : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584CCE : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584CDE : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584CEE : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584CFE : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D0E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D1E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D2E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D3E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D4E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D5E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
0x01584D6E : 0000 0000 0000 0000 0000 0000 0000 0000    .....
```

2006-03-07-18.13.01.511000-480 I401046H451

LEVEL: Warning

PID : 2232

TID : 2252

PROC : db2bp.exe

INSTANCE: DB2

NODE : 000

APPID : G92B7027.I304.011848021301

FUNCTION: DB2 UDB, DRDA Application Requester, sqljrListToConnect, probe:20

MESSAGE : Reconnecting to Service name/Port number -->

DATA #1 : Hexdump, 15 bytes

```
0x01584D7E : 3530 3030 3000 0000 0000 0000 0000 00    50000.....
```

2006-03-07-18.13.01.912000-480 I401499H346

LEVEL: Warning

PID : 2232

TID : 2252

PROC : db2bp.exe

INSTANCE: DB2

NODE : 000

APPID : G92B7027.I404.011848021303

FUNCTION: DB2 UDB, DRDA Application Requester, sqljrClientReroute, probe:998

MESSAGE : Client Reroute is completed successfully.

Example 4-118 Connection state from the client machine

DB2 GET CONNECTION STATE

Database Connection State

Connection state = Connectable and Connected

Connection mode = SHARE

Local database alias = SOURCE

Database name = SOURCE

Hostname = denmark

Service name = 50000

DB2 LIST DB DIRECTORY

System Database Directory

Number of entries in the directory = 4

Database 1 entry:

| | |
|-------------------------------------|------------------|
| Database alias | = SOURCE |
| Database name | = SOURCE |
| Node name | = DENMARK |
| Database release level | = a.00 |
| Comment | = |
| Directory entry type | = Remote |
| Catalog database partition number | = -1 |
| Alternate server hostname | = jamaica |
| Alternate server port number | = 50000 |

.....

4.8.7 STEP TESTREVERT7: Verify Q replication up and running

In this step, verify that the unidirectional Q replication environment is operating normally by checking the Q Capture process on DENMARK, the subscription states on the primary server DENMARK and the secondary servers CLYDE and BONNIE using scripts and commands similar to that described in 4.6.9, “STEP TESTPTKO9: Verify Q replication up and running” on page 178.

4.8.8 STEP TESTREVERT8: Verify all workload changes applied to target tables

In this step, verify that all the changes that have been applied on JAMAICA and DENMARK (after reverting back to the original configuration) by the application workload have been successfully applied to the secondary servers CLYDE and BONNIE.

Example 4-120 show the SQL to connect to the primary server DENMARK and list the contents of the PRODUCTS and SALES table, while Example 4-121 on page 220 and Example 4-122 on page 221 show the SQL to view the contents of the corresponding tables on the secondary servers CLYDE and BONNIE respectively.

Attention: The content of the SOURCE database on the primary server DENMARK fully matches the corresponding contents on the secondary servers CLYDE and BONNIE — this indicates that HADR and unidirectional replication coexist synergistically and seamlessly in reverting back to the original HADR environment where the primary server is DENMARK and the standby server is JAMAICA.

Example 4-120 Contents of source tables on the primary server DENMARK

CONNECT TO SOURCE

Database Connection Information

Database server = DB2/AIX64 9.1.0
 SQL authorization ID = DB2INST9
 Local database alias = SOURCE

SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|---------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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 DB2INST9.SALES WHERE REGION_CODE = 'EAST' ORDER BY REGION_CODE, PRODUCT_ID

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | REGION_CODE |
|------------|----------|---------------|--------------|-------------|
| 10 | 100 | 1000 | 03/01/2006 | EAST |
| 20 | 120 | 2000 | 03/01/2006 | EAST |
| 30 | 100 | 5000 | 03/02/2006 | EAST |
| 60 | 160 | 6000 | 03/03/2006 | EAST |

4 record(s) selected.

```
SELECT * FROM DB2INST9.SALES WHERE REGION_CODE = 'WEST' ORDER BY REGION_CODE,
PRODUCT_ID
```

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE | REGION_CODE |
|------------|----------|---------------|--------------|-------------|
| 10 | 200 | 0 | 03/02/2006 | WEST |
| 20 | 220 | 2000 | 03/02/2006 | WEST |
| 30 | 200 | 3000 | 03/03/2006 | WEST |
| 30 | 200 | 5000 | 03/01/2006 | WEST |
| 40 | 140 | 4000 | 03/02/2006 | WEST |
| 40 | 240 | 6000 | 03/01/2006 | WEST |
| 80 | 270 | 5 | 03/04/2006 | WEST |

7 record(s) selected.

Example 4-121 Contents of target tables on the secondary server CLYDE

CONNECT TO TARGETA

Database Connection Information

Database server = DB2/AIX64 9.1.0
 SQL authorization ID = DB2INST9
 Local database alias = TARGETA

```
SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID
```

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|---------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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 DB2INST9.SALES_EAST ORDER BY PRODUCT_ID
```

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE |
|------------|----------|---------------|--------------|
| 10 | 100 | 1000 | 03/01/2006 |
| 20 | 120 | 2000 | 03/01/2006 |
| 30 | 100 | 5000 | 03/02/2006 |
| 60 | 160 | 6000 | 03/03/2006 |

4 record(s) selected.

Example 4-122 Contents of target tables on the secondary server BONNIE

CONNECT TO TARGETB

Database Connection Information

Database server = DB2/AIX64 9.1.0
SQL authorization ID = DB2INST9
Local database alias = TARGETB

SELECT * FROM DB2INST9.PRODUCTS ORDER BY PRODUCT_ID

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRICE | MANUFACT_ID |
|------------|------------------|------------------|---------|-------------|
| 10 | WIDGETS | TOOLS | 1.00 | 100 |
| 20 | THINGAMABOBBER | 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 DB2INST9.SALES_WEST ORDER BY PRODUCT_ID

| PRODUCT_ID | STORE_ID | QUANTITY_SOLD | DATE_OF_SALE |
|------------|----------|---------------|--------------|
| 10 | 200 | 0 | 03/02/2006 |
| 20 | 220 | 2000 | 03/02/2006 |
| 30 | 200 | 5000 | 03/01/2006 |
| 30 | 200 | 3000 | 03/03/2006 |
| 40 | 140 | 4000 | 03/02/2006 |
| 40 | 240 | 6000 | 03/01/2006 |
| 80 | 270 | 5 | 03/04/2006 |

7 record(s) selected.



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 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:

<ftp://www.redbooks.ibm.com/redbooks/SG247216/>

The code and scripts are packaged as follows:

1. BIDI.zip file that includes the code and scripts to set up the bidirectional Q replication environment.

A BIDIREADME.txt file is associated with this zip file that lists its contents with a brief description of each of the scripts provided.

2. HADRUNIDIR.zip file that includes the code and scripts to set up the HADR high availability unidirectional Q replication environment.

A HADRUNIDIRREADME.txt file is associated with this zip file that lists its contents with a brief description of each of the scripts provided.

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 schema¹ 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. |

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 |
|-------------|-------------------|---|
| REASON | CHAR(12) NOT NULL | <p>A description of the error or conflict that caused the transaction to be logged into the IBMQREP_EXCEPTIONS table.</p> <ul style="list-style-type: none"> ▶ “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 in the list of acceptable errors in the OKSQLSTATES column of the IBMQREP_TARGETS table. ▶ “OKSQLSTATE” An SQL error occurred, and it was in 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 didn’t 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. |
| SQLSTATE | CHAR(5) | The SQL state number returned by DB2 for the transaction. |

| Column name | Data type | Description |
|-------------|-----------------------|---|
| 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: <ul style="list-style-type: none"> ► “Y” The row was applied because the CONFLICT_ACTION specified for the Q subscription was “F” (force). ► “N” The transaction was not applied. |

| Column name | Data type | Description |
|---------------|---|---|
| CONFLICT_RULE | CHAR(1) | The type of conflict detection that resulted in the row being entered in the IBMQREP_EXCEPTIONS table. Values are: <ul style="list-style-type: none"> ► “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 identify not 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.

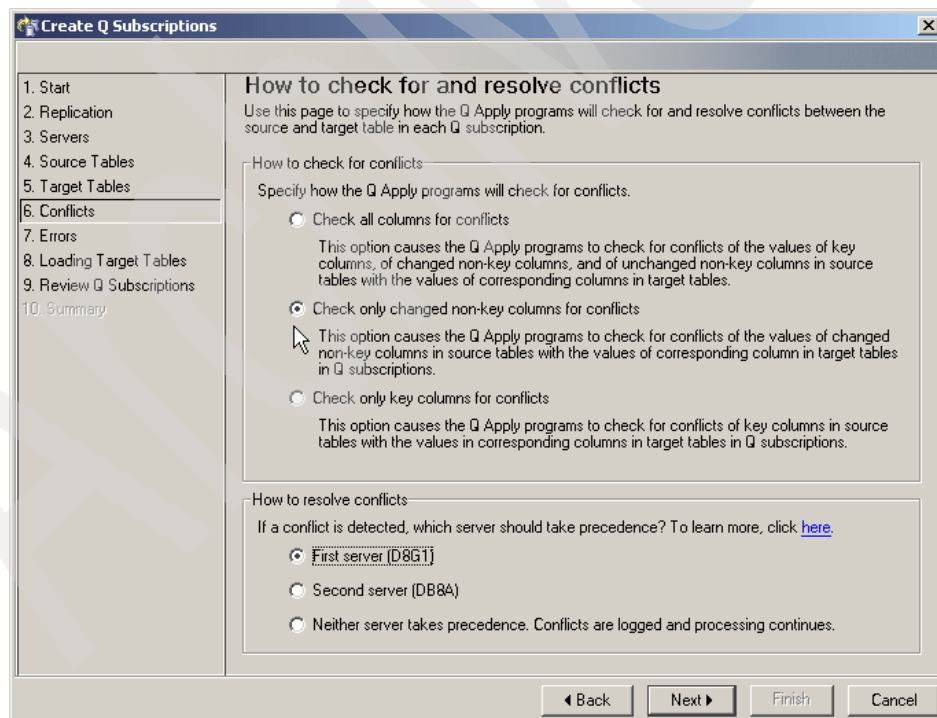


Figure B-1 Conflict rule for a subscription in bidirectional Q replication

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 into 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 on page 232 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 B.1.1.2, “Conflict action” on page 233.

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 is specifiable as shown in Figure B-2 on page 233.

Create Q Subscriptions

1. Start
2. Replication
3. Servers
4. Source Tables
5. Target Tables
6. Conflicts
7. Errors
8. Loading Target Tables
9. Review Q Subscriptions
10. Summary

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 (DBG1)
- ☐ Neither server takes precedence. Conflicts are logged and processing continues.

◀ Back Next ▶ Finish Cancel

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 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 when 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.

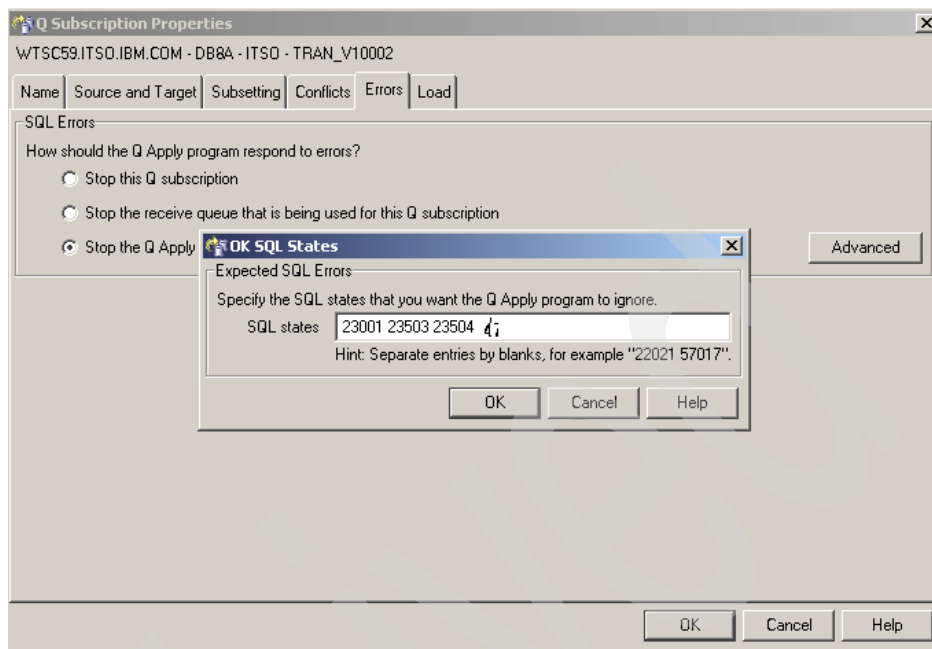


Figure B-3 Specifying OKSQLSTATES for subscription in Replication Center

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 B.1.2.1, “Error action” on page 237.

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 on page 238. 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.

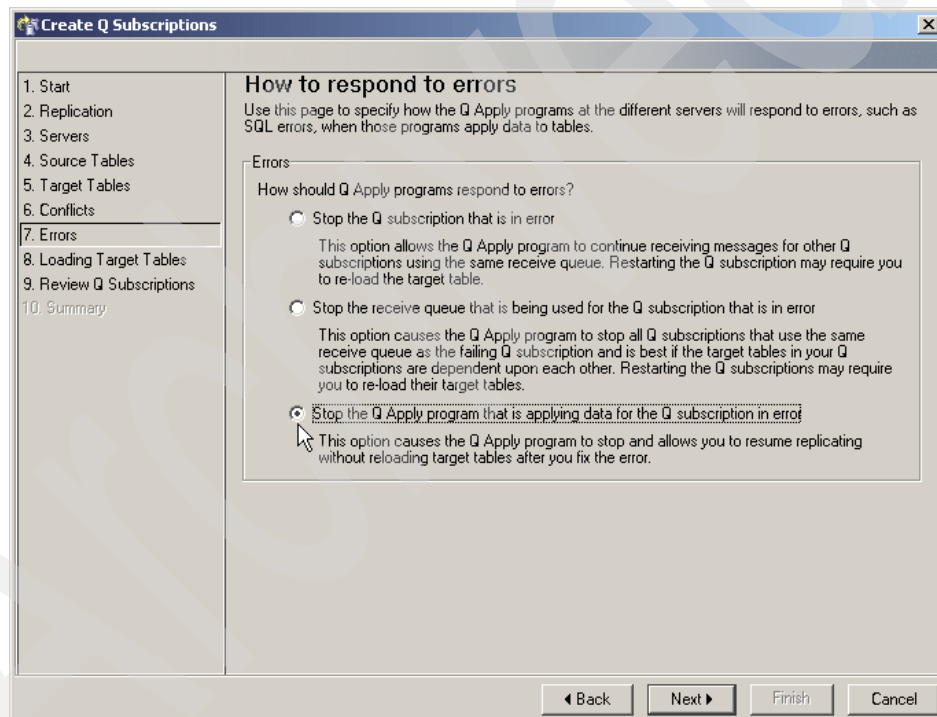


Figure B-4 Error action for a subscription - both bidirectional and peer-to-peer replication

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 B.1.1.2, "Conflict action" on page 233.

- 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 B.1.1.2, "Conflict action" on page 233.

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 B.1.1.2, "Conflict action" on page 233.

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 on page 240 shows an example of invoking this tool from the command line.

Example B-5 on page 240 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 if any — 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 ITS0 -uid hueykim -pwd
xxxxxxx
```

```
Exception Time      = 2006-02-28 14:17:24.111371
RECVC               = QREP.POKA.TO.POKB.REVCQ
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) = ??????~???????~?""
                   (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 on page 241 through Figure B-9 on page 244.

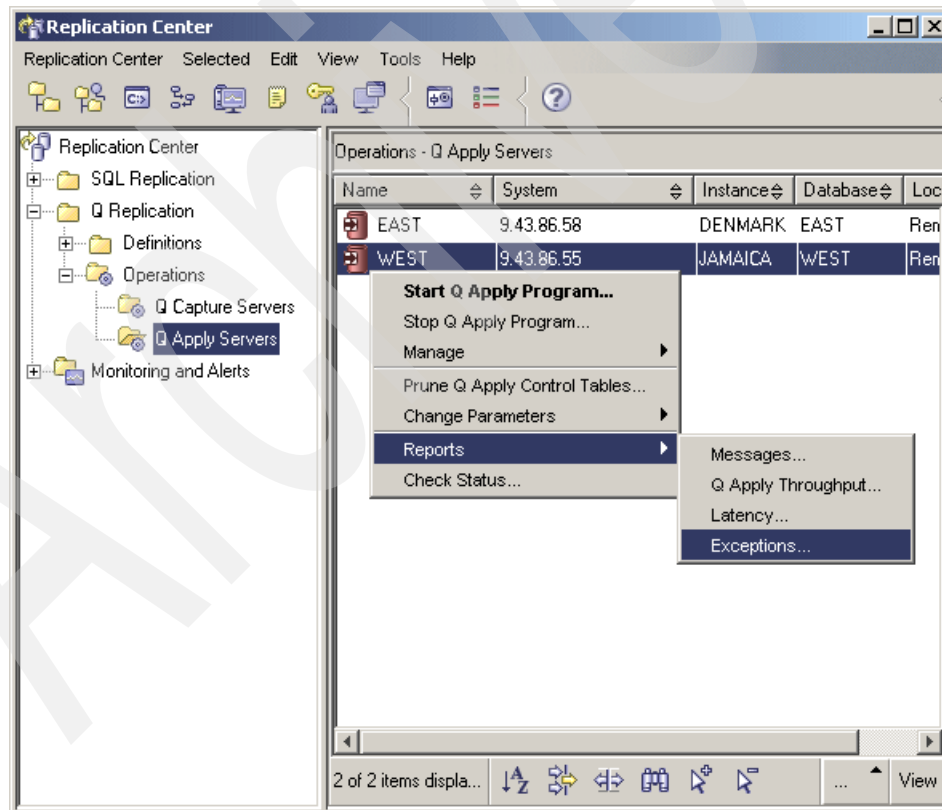


Figure B-5 Using Replication Center to view exceptions 1/5

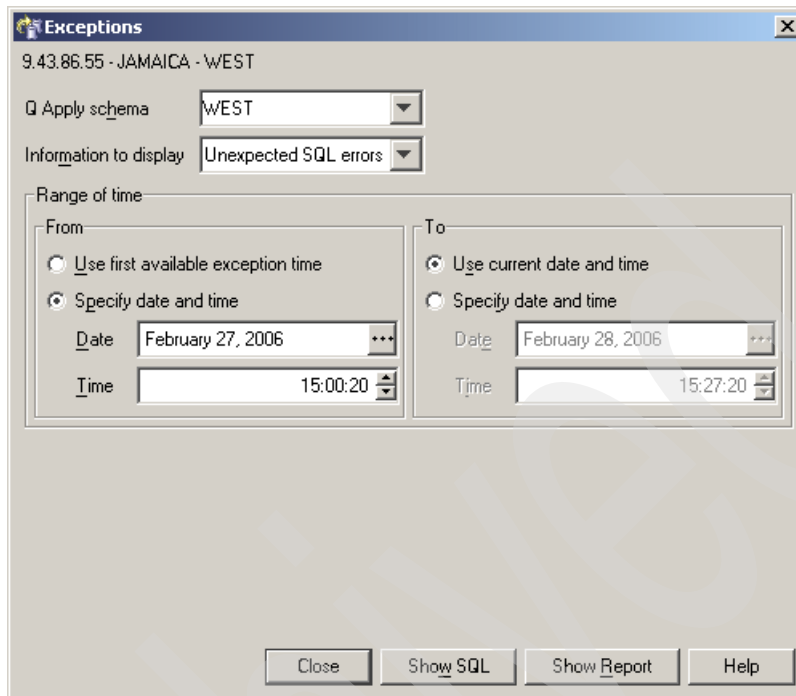


Figure B-6 Using Replication Center to view exceptions 2/5

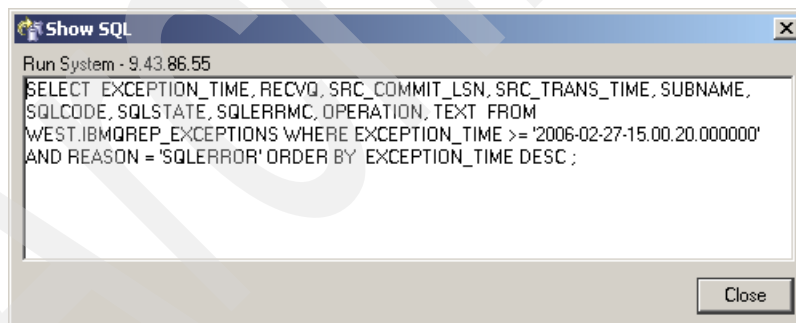


Figure B-7 Using Replication Center to view exceptions 3/5

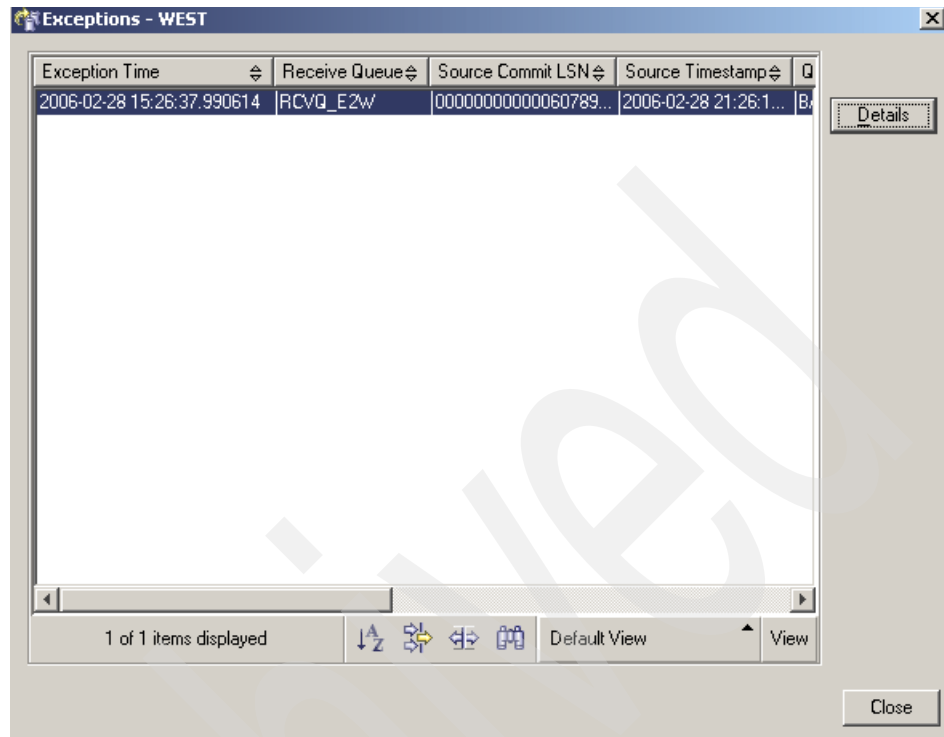


Figure B-8 Using Replication Center to view exceptions 4/5

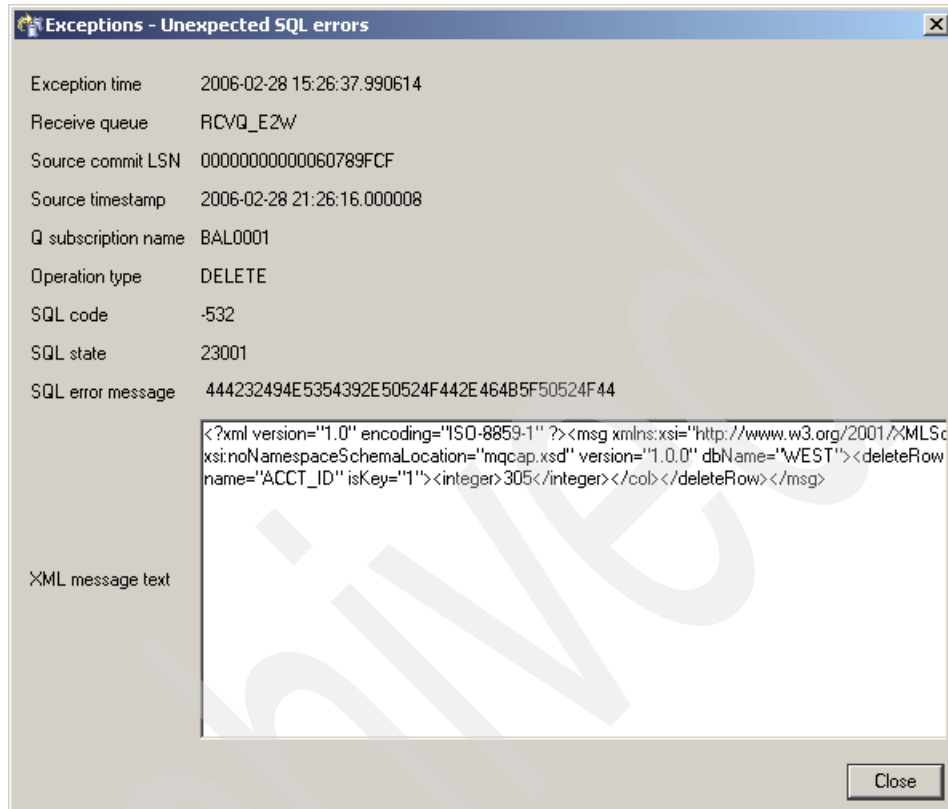


Figure B-9 Using Replication Center to view exceptions 5/5

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 on page 245 and Figure B-11 on page 246 show how the contents of the exceptions table may be viewed via the Live Monitor tool.

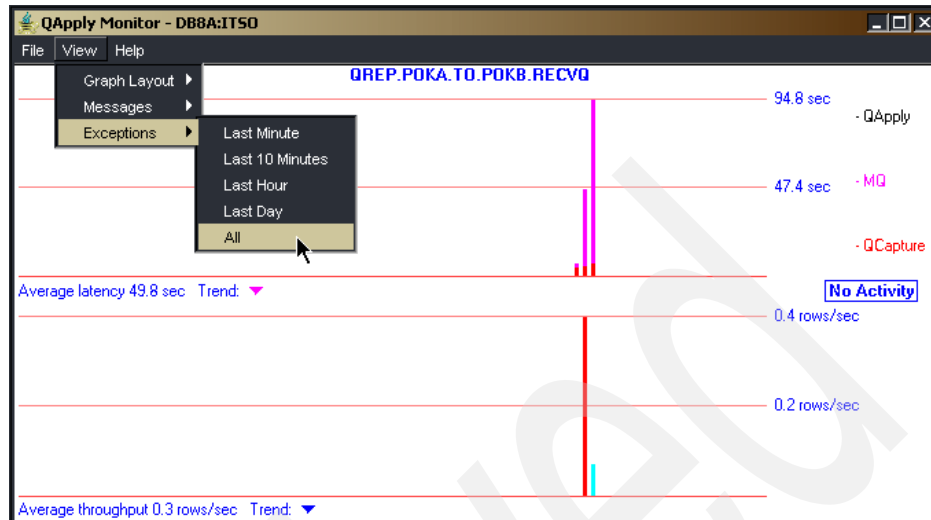


Figure B-10 Using Live Monitor to view exceptions 1/2

| 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 HA scenarios

As described in B.1.1, “Conflict considerations” on page 230, 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 B.3, “Data inconsistencies in HA bidirectional scenario” on page 276.

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 some 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 on page 248. 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.

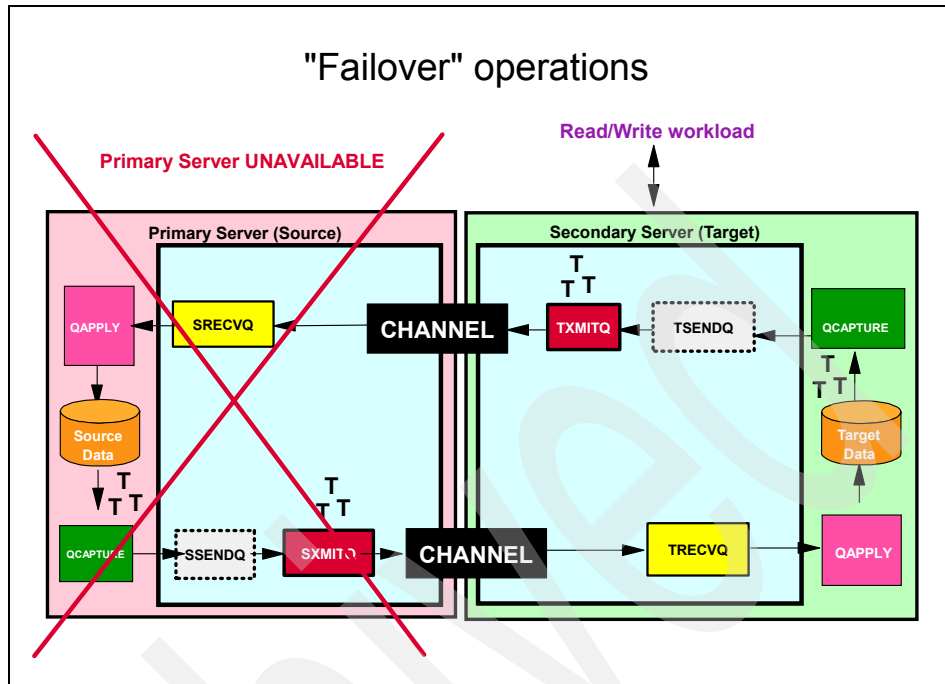


Figure B-13 Bidirectional Q replication HA environment - failover operations

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 on page 250, 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.

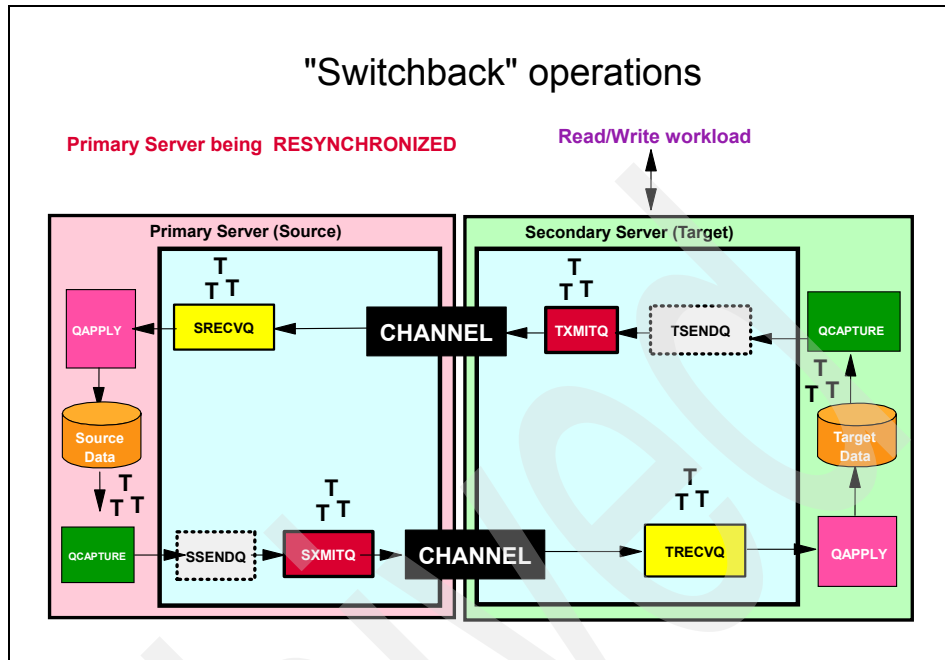


Figure B-14 Bidirectional Q replication HA environment - switchback operations

Figure B-15 on page 251 through Figure B-17 on page 253 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 on page 251 through Figure B-17 on page 253, the following definitions apply:

- ▶ Independent row is a row in a table which 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 constraints 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.
- ▶ Parent row is a row in a table which has referential constraints defined, and the row have child rows associated with it.
- ▶ Child row is a row in a table which has referential constraints defined, and the row has one or more² parent rows associated with it.

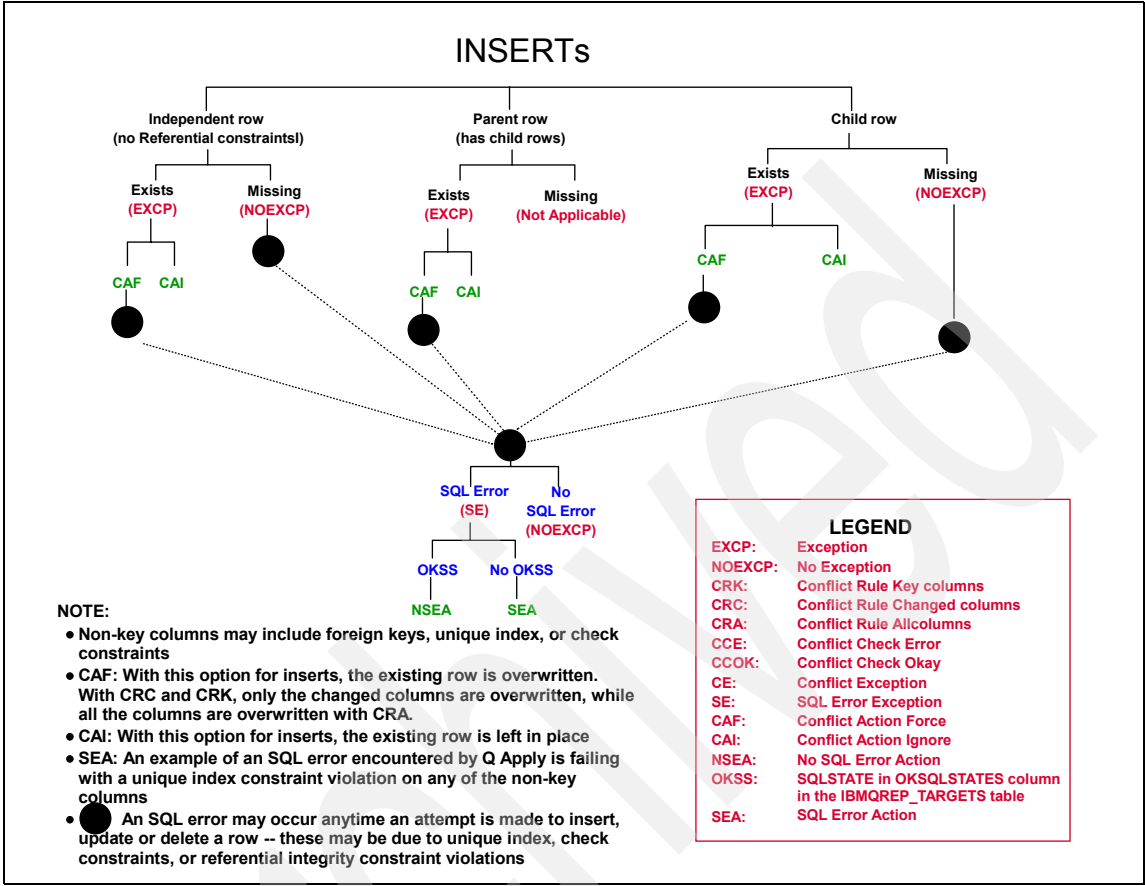


Figure B-15 Propagated INSERTs processing

² When a row has multiple non-null foreign keys associated with it.

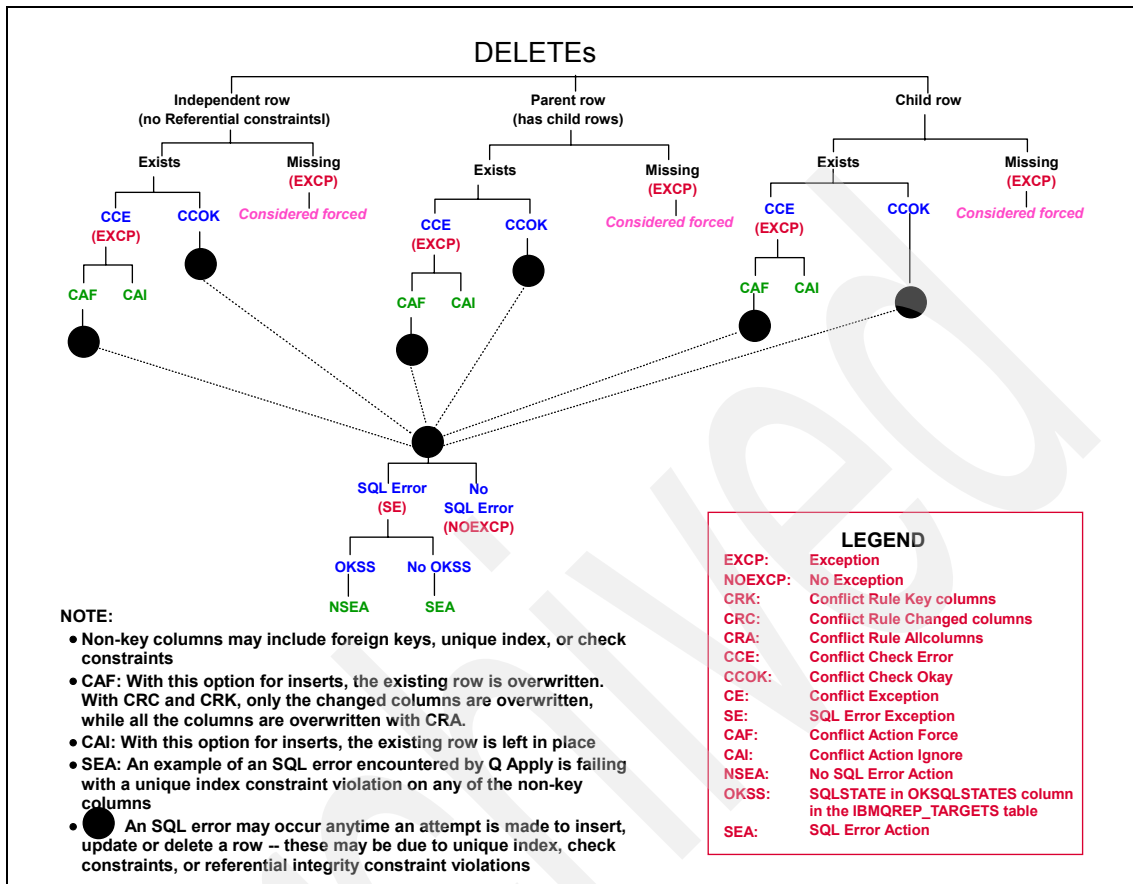


Figure B-16 Propagated DELETes processing

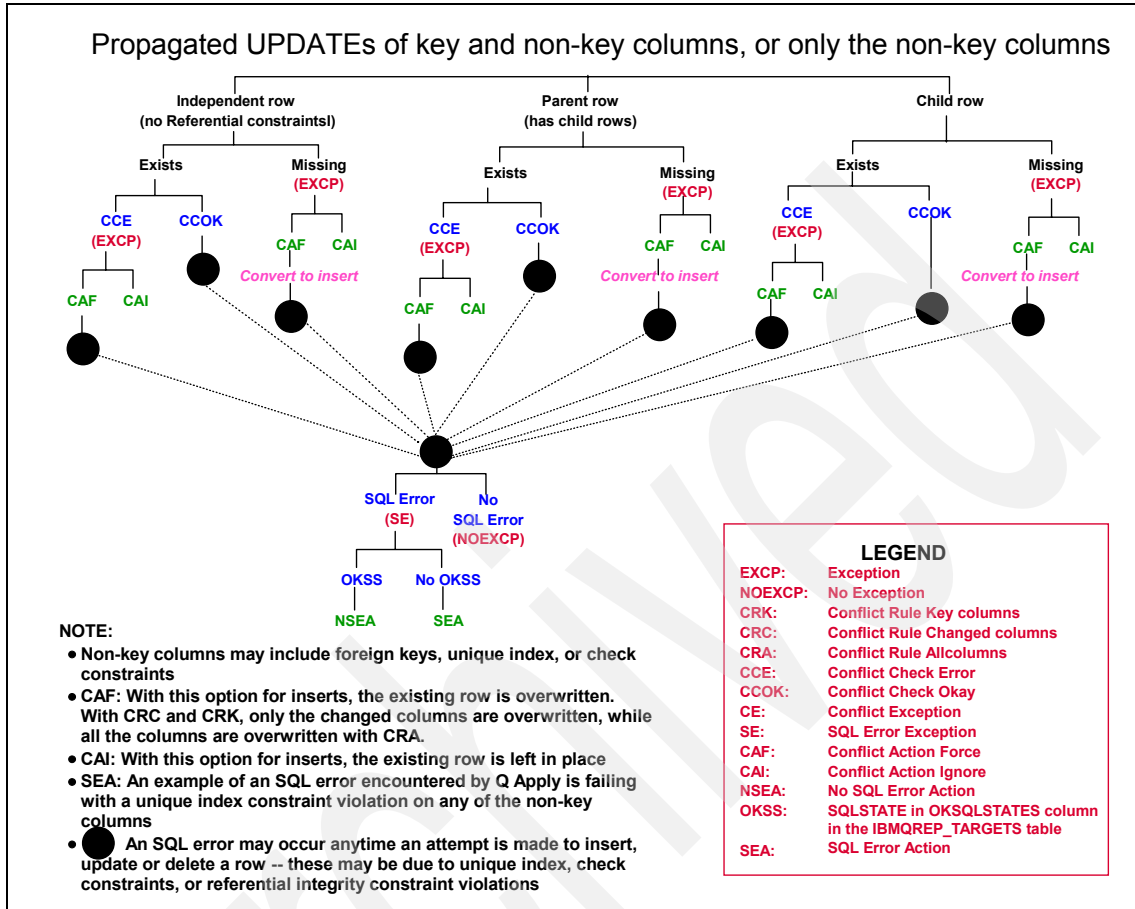


Figure B-17 Propagated UPDATES processing (key and non key columns, or only the non key columns)

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. The following conflicts are described:

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 though the existing row were an independent row, regardless of whether the delete rule is CASCADE or RESTRICT for the dependent table(s).

Example: B-7 Insert the 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) = ????"?~??????"
               (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 254 is that the conflict action “IS_APPLIED” is ‘N’ which corresponds to an ignore.

Example: B-8 Insert the 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) = ????"?~??????"
               (HEX) = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F2F0F6

```

```
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 same key and child's parent(s) 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 on page 257 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 table(s).

Example: B-9 Insert the child row and the 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) | = C2C1D3D9E5F1D9D7FFF0F0F0F0F0F0F2F1C5 |
| 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 with Example B-9 is that the conflict action “IS_APPLIED” is ‘N’ which corresponds to an ignore.

Example: B-10 Insert the child row and the same key and the child's parent(s) 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 on page 259 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??Zm†?'
               (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 and same key exists, child parent(s) missing

This scenario corresponds to the case where a propagated row that is a child row finds a row with the corresponding key³, 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 conflict 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 follows:

- 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 or not the propagated row was actually applied successfully or not — the correlating fields are SRC_COMMIT_LSN and SRC_TRANS_TIME.

Example: B-12 Insert the child row but the same key and child's parent is missing - FORCE 1/2

| | |
|----------------|------------------------------|
| Exception Time | = 2006-03-07 22:18:24.325188 |
| RECVQ | = QREP.POKB.TO.POKA.RECVQ |
| SRC_COMMIT_LSN | = x'00000000259D7AF10000' |

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

```

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?¼
                   (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 the child row but the 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 with Example B-12 on page 260 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) = ??Z?+??~???????o?
                  (HEX)  = D7D9D6C4D9E5F1D9FFF0F0F0F0F0F0F2F0F1
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. The following conflicts are described:

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 with Example B-15 on page 263 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
RECVC              = 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 on page 265 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 with Example B-17 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 or not the propagated row was actually applied successfully or not — 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) = C6D26DD7D9D6C46DE5F1FFF0F0F0F0F0F0F2F0F2
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 with Example B-19 on page 267 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 the parent row and the same key exists - value-based conflict - IGNORE

```

Exception Time   = 2006-03-07 22:43:13.8796
RECVC           = 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. The following conflicts are described:

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 or not the propagated row was actually inserted successfully or not — 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 |
| RECVC | = 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 269 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 indep./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 follows:

- 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 or not the propagated row was actually applied successfully or not — 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 with Example B-24 on page 272 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 indep./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.

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 such as referential constraint or uniqueness constraint violations.

Example B-27 on page 275 shows the output of the Exception Table Formatter for an SQL error exception that occurs without a corresponding conflict exception — the error encountered was 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 the 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 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 on page 278 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 278.
- ▶ 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 278.
- ▶ 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 291

As mentioned earlier, for the bidirectional high availability scenario described in this redbook, 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, 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.

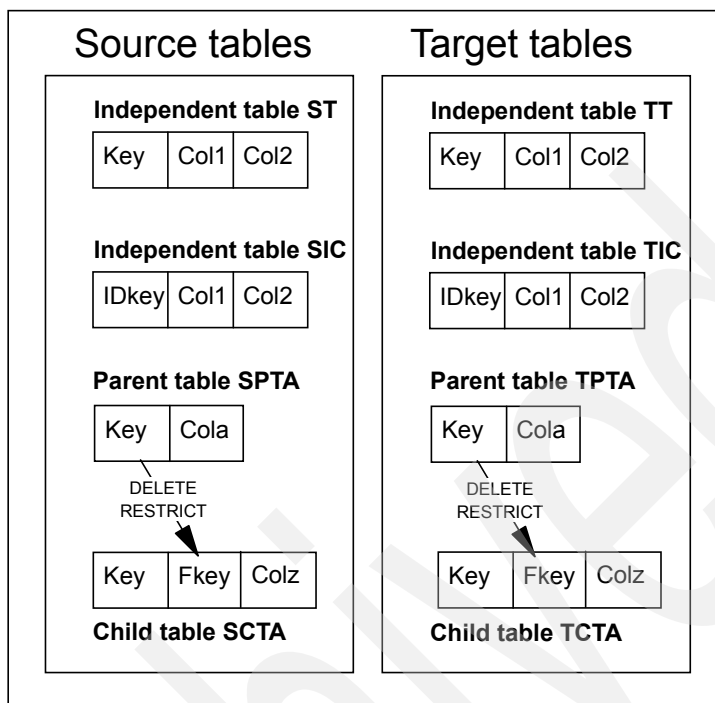


Figure B-18 Tables used to demonstrate the data inconsistency scenarios

B.3.1 Non-referential integrity related data inconsistency scenarios

Table B-2 on page 279 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 switch-back completed | Source exceptions <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATES 3001 23503 23504 | Target exceptions <ul style="list-style-type: none"> ► 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 'Insert' 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 |
| 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' |

| S# | Unpropagated changes before failover | Unpropagated changed before switchback | Source data after switchback completed | Target data after switchback completed | Source exceptions | Target exceptions |
|----|---|--|--|--|---|--|
| | | | | | <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATES 3001 23503 23504 | <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATES 23001 23503 23504 |
| 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 on page 279 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 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 the 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 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 are summarized in Table B-2 on page 279 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 — lets 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 which 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. Lets 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 B.3.1, “Non-referential integrity related data inconsistency scenarios” on page 278.

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 | Target exceptions |
|----|--|--|--|--|---|--|
| | | | | | <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATE 001 23503 23504 | <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATE 3001 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' |

| S# | Unpropagated changes before failover | Unpropagated changed before switchback | Source data after switchback completed | Target data after switchback completed | Source exceptions | Target exceptions |
|----|---|--|--|--|--|--|
| | | | | | <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATE 23001 23503 23504 | <ul style="list-style-type: none"> ► Conflict rule is 'C' ► Conflict action is FORCE ► Error action is Q ► OKSQLSTATE 3001 23503 23504 |
| 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 are summarized in Table B-3 on page 292 are briefly described here:

1. Scenario 1

- 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.
- 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.
- 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 row 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 row 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 row 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 row 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) is 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.

Overview of HADR and DB2 Client Reroute

In this appendix we provides a brief overview of HADR and DB2 Client Reroute.

C.1 HADR

DB2 Universal Databases (DB2 UDB) high availability disaster recovery (HADR) is a database replication feature that provides a high availability solution for both partial and complete site failures. HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby.

Figure C-1 provides a high level overview of HADR.

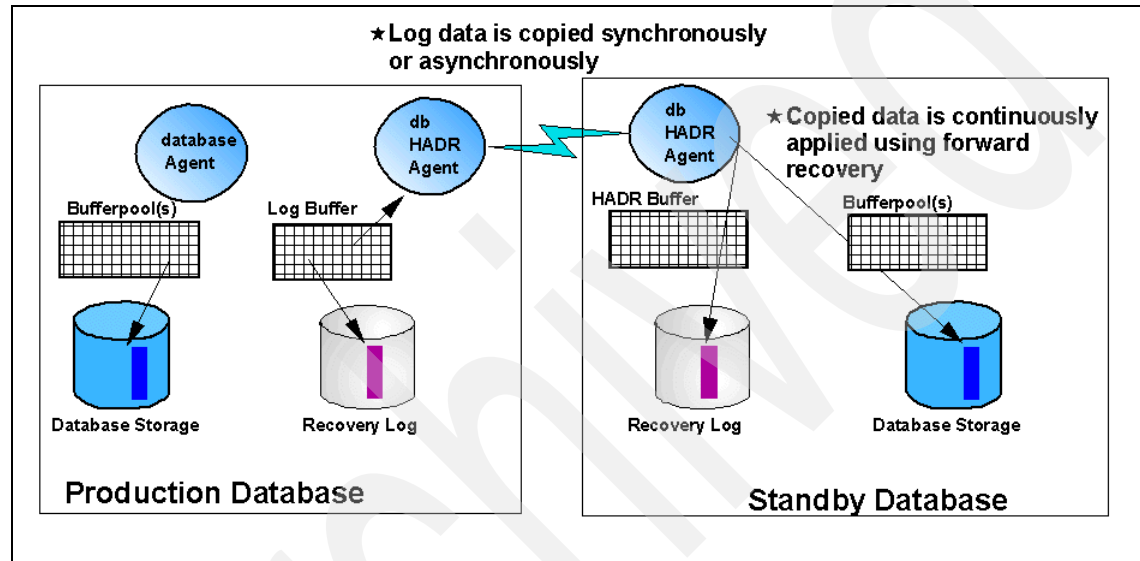


Figure C-1 HADR overview

HADR relies on the DB2 log as the single source where all changes made to a specific database are recorded. By shipping the log records from the production DB2 server (called the primary server), to a backup DB2 server (called the standby server), all the changes made on the primary server are now available at the standby server. The standby server simply needs to replay each log record to its copy of the DB2 database to keep the backup database synchronized with the primary database.

With high availability disaster recovery (HADR), you can specify one of three synchronization modes to choose your preferred level of protection from potential loss of data. The synchronization mode indicates how log writing is managed between the primary and standby databases. These modes apply only when the primary and standby databases are in peer state. Use the `HADR_SYNCMODE` configuration parameter to set the synchronization mode. The different synchronization modes are shown in Figure C-2 on page 303.

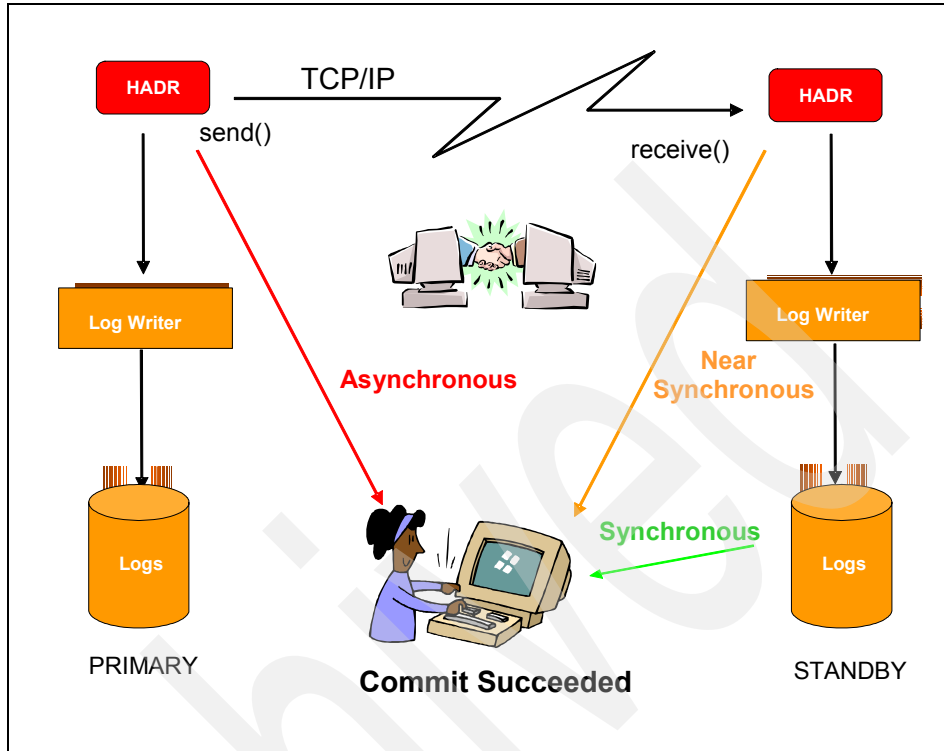


Figure C-2 Different synchronization modes of HADR

The valid values for the HADR_SYNCMODE configuration parameter are as follows:

- ▶ SYNC (synchronous) mode provides the greatest protection against transaction loss, and using it results in the longest transaction response time among the three modes. In this mode, log writes are considered successful only when logs have been written to log files on the primary database and when the primary database has received acknowledgement from the standby database that the logs have also been written to log files on the standby database. The log data is guaranteed to be stored at both sites.
- ▶ NEARSYNC (near synchronous) mode has a shorter transaction response time than synchronous mode, but it also provides slightly less protection against transaction loss. In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and when the primary database has received acknowledgement from the standby system that the logs have also been written to main memory on the standby system. Loss of data occurs only if both sites fail

simultaneously and if the target site has not transferred to nonvolatile storage all of the log data that it has received.

- ▶ ASYNC (asynchronous) mode has the highest chance of transaction loss if the primary system fails. It also has the shortest transaction response time among the three modes. In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and have been delivered to the TCP layer of the primary system's host machine. Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby. A failure on the primary database host machine, on the network, or on the standby database can cause log files in transit to be lost. If the primary database is available, the missing log files can be resent to the standby database when the pair reestablishes a connection. However, if a failover operation is required while there are missing log files, both the log files and the associated transactions will never reach the standby database. Permanent loss of transactions is caused by lost log files and a failure on the primary database. If transactions are lost, the new primary database is not exactly the same as the original primary database after a failover operation.

Note: HADR might be your best option if most or all of your database requires protection, or if you perform DDL operations that must be automatically replicated on the standby database.

Applications can only access the current primary database.

A partial site failure can be caused by a hardware, network, or software (DB2 or operating system) failure. Without HADR, a partial site failure requires the database management system (DBMS) server or the machine where the database resides to be rebooted. The length of time it takes to restart the database and the machine where it resides is unpredictable. It can take several minutes before the database is brought back to a consistent state and made available. With HADR, the standby database can take over in seconds. Furthermore, you can redirect the clients that were using the original primary database to the standby database (new primary database) by using automatic client reroute as described in “DB2 Client Reroute” on page 305 or retry logic in the application.

A complete site failure can occur when a disaster, such as a fire, causes the entire site to be destroyed. Because HADR uses TCP/IP for communication between the primary and standby databases, they can be situated in different locations. For example, your primary database might be located at your head office in one city, while your standby database is located at your sales office in another city. If a disaster occurs at the primary site, data availability is maintained

by having the remote standby database take over as the primary database with full DB2 functionality. After a takeover operation occurs, you can bring the original primary database back up and return it to its primary database status; this is known as failback.

Note: For further details on HADR, refer to *IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference Version 8.2*, SC09-4831.

C.2 DB2 Client Reroute

When a DB2 Universal Database™ (DB2 UDB) client application suffers from a loss of communication with a DB2 UDB server, you want the client to be able to recover from such a loss without any intervention by you or another administrator.

The automatic client reroute feature allows client applications to recover from a loss of communication with the server so that they can continue to work with minimal interruption. After a loss of communication, the client application attempts to reconnect to the server. If this fails, the client is then automatically rerouted to an alternate server as shown in Figure C-3.

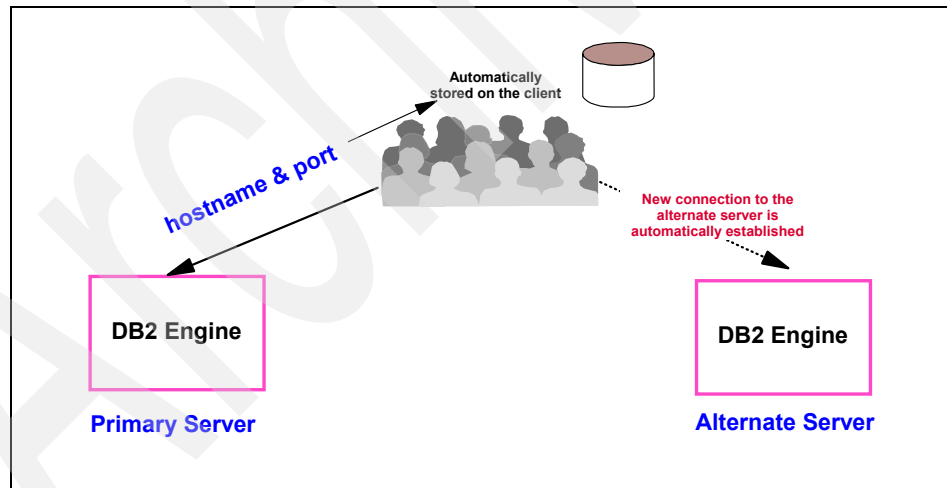


Figure C-3 DB2 Client Reroute

Prior to the communication failure, you need to establish an alternative location that the client connection knows. The **UPDATE ALTERNATE SERVER FOR DATABASE** command is used to define the alternate server location on a particular database. The alternate hostname and port number is given as part of the command. The location is stored in the system database directory file at the server.

After you have specified the alternate server location on a particular database at the server instance, the alternate server location information is returned to the client as part of the connection process. If communication between the client and the server is lost for any reason, the DB2 UDB client code will attempt to re-establish the connection by using the alternate server information. The DB2 UDB client will attempt to re-connect with the original server and the alternate server, alternating the attempts between the two servers. The timing of these attempts varies from very rapid attempts to begin with gradual lengthening of the intervals between the attempts. Once a connection is successful, the SQLCODE -30108 is returned to indicate that a database connection has been re-established following the communication failure. The hostname/IP address and service name/port number are returned. The client code only returns the error for the original communications failure to the application if the re-establishment of the client communications is not possible to either the original or alternative server.

Note: For further details on DB2 Client Reroute, refer to *IBM DB2 Universal Database Administration Guide: Implementation Version 8.2*, SC09-4820.

Important: The automatic client reroute feature can be used with HADR to allow client applications to recover from a loss of communication with the server and to continue working with minimal interruption. Automatic Client Rerouting is only possible when an alternate database location has been specified at the server. Automatic client reroute is only supported with TCP/IP protocol. You can use automatic client reroute with HADR to make client applications connect to the new primary database after a takeover operation. If automatic client reroute is not enabled, client applications will receive error message SQL30081, and no further attempts will be made to establish a connection with the server.

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/SG247216>

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, SG247216.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

| <i>File name</i> | <i>Description</i> |
|-----------------------|---|
| BIDI.zip | 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. |
| HADRUNIDIR.zip | Code and scripts to set up the HADR high availability unidirectional Q replication environment, and perform the steps for planned takeover, unplanned takeover, and reestablishing the original environment after a planned or unplanned takeover has occurred. |

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 (folder) 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 310. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *MQSeries Primer*, REDP-0021
- ▶ *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487
- ▶ *WebSphere MQ V6 Fundamentals*, SG24-7128

Other publications

These publications are also relevant as further information sources:

- ▶ “The IBM DB2 Version 8.2 Automatic Client Reroute Facility”, by Paul C. Zikopoulos, December 2005 at:
<http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0512zikopoulos/>
- ▶ *IBM WebSphere for Replication Server, Replication and Event Publishing Guide and Reference Version 8.2*, SC18-7568
- ▶ *IBM WebSphere for Replication Server, ASNCLP Program Reference for Replication and Event Publishing Version 8.2*, SC18-9410
- ▶ *IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference Version 8.2*, SC09-4831
- ▶ *IBM DB2 Universal Database Administration Guide: Implementation Version 8.2*, SC09-4820
- ▶ *WebSphere MQ for AIX, Quick Beginnings Version 5.3*, GC34-6076
- ▶ *WebSphere MQ, System Administration Guide*, SC34-6068
- ▶ *WebSphere MQ, Script (MQSC) Command Reference*, SC34-6055
- ▶ *WebSphere MQ, Security Version 5.3*, SC34-6079

- ▶ *WebSphere MQ, Intercommunication*, SC34-6059

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/qrep1-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

A

Active/Active 5, 247, 276
Active/Passive 5
Administration queue 24
agent threads 26–27, 32
analyze and resolve all exceptions 80, 95, 105
application workload 51, 56, 64–65, 123, 129, 133, 156
ASNCLP commands 14, 151–153
asynchronous 11, 127, 131, 304
automatic load 72, 95–96, 102, 106, 155, 238
automatic loading 18, 155
automatically resync primary and secondary servers 76
Autostop Q Apply 55, 73, 87, 105
Autostop Q Capture 51, 65, 79, 87, 104
Autostop Q Capture on secondary 113
availability xix–xx, 1–5, 28, 34, 39, 49, 71, 117–121, 224, 230, 234, 277, 302, 304, 308
 levels 2–3

B

basic systems 4
bidirectional replication xix, 12, 29, 33–37, 50, 230, 232–235, 246
bidirectional replication topology 36, 39
browser thread 26–27
build messages 187
business requirement 34, 117–118

C

CAPSTART signal 15
CCD 12
CF 50–51, 56–57
channels 12, 25, 39, 114, 144
code and scripts xix, 223–224
Cold start Q Capture on both servers 100, 108, 112
COMMIT_INTERVAL 21, 31
compact format 11, 21
component failover 4
conflict action 253–257
conflict action of force 253, 260

conflict action of ignore 262
conflict considerations 229
conflict detection 28, 227, 229, 232, 235, 246
conflict rule 235, 253, 277, 279, 292
CONFLICT_RULE 83–84, 229, 231–232, 240
conflicts 26, 28, 35, 42–43, 49, 66, 226, 231–232, 234
consistent change data 12
continuous availability 7
continuous operation 3, 7
controlled failover 44, 49–50
controlled failover and switchback 33–34, 49
Create Q subscriptions 153
Create replication queue maps 153
CURRENT_LOG_TIME 30–31, 60

D

data inconsistencies 225, 276
data loss 2, 6, 35, 42, 44–47, 156, 185, 198, 302
DB2 Client Reroute 120–122, 134–135, 301, 306
DB2 log 42–45, 52, 121–122, 172, 187, 280, 302
DB2 log retention 44
DB2 recovery log 10, 19, 23–24, 30
db2diag.log 140, 164, 166, 168, 174
deactivate subscriptions 96, 98–99, 106, 108
delete messages in relevant queues 88, 105
dependency analysis 25
diagnostic log 27
disaster recovery 6
disaster recovery scenario 47, 109, 114
discard primary changes and reinitialize primary 109
disk mirroring 4
DLQ 26

E

end-to-end latency 21, 32, 44, 50, 60, 62–63
ensure no exceptions recorded 67
environment configuration 33–34, 36, 117–119
error action 238, 253, 277, 279, 292
ERROR_ACTION 43, 239
exceptions 36, 46, 48, 50, 56, 80, 95, 105, 126, 130, 225–226, 229, 239–241, 247

Exceptions Table Formatter 239
exceptions with deletes 254, 263
exceptions with inserts 254
exceptions with updates 254, 269

F

failback 198–199, 305
failover xix, 4–5, 28, 33–35, 41–42, 127, 132, 139,
247–249, 276–277, 304, 308
failover processing considerations 41
fallback 207
fault detection time 2
FIFO 26, 149
filtering 12–13, 20
firewalls 4
five nines 6

H

HA 2, 7, 118, 120, 144, 225, 246–249
HADR xix, xxi, 2, 117–121, 224, 301–305, 308
HADR_SYNCMODE 124–125, 127, 132, 302–303
high availability xix, xxi, 1–3, 7, 118, 120–121, 235,
302
High Availability Disaster Recovery 121, 166–169
hot site back up 34

I

ignore conflicts 234

L

latency 10–11, 19–21, 34, 50, 59, 62–63, 118–119
latency considerations 9, 29
latency statistics 30, 44, 60–61
Live Monitor 60, 239, 244–246
load balancers 4
LOB 20, 26, 28, 44, 229, 232
low-latency 10

M

manual load 17–18
MAX_MESSAGE_SIZE 31
MAXDEPTH 25, 43, 149
MAXMSGL 25, 148–149
memory limit 28
MEMORY_LIMIT 31–32
monitor information 28
MONITOR_TIME 30–31, 60

MTBF 2–3
MTTR 2–3
multi-master replication 13

N

NUM_APPLY_AGENTS 32

O

one-way replication 28
Option A 46, 48, 76–77
Option B 46, 84, 86
Option C 46, 102, 104
Option D 47, 109, 111

P

parallel 25–27, 32, 168, 203, 212
peer-to-peer replication 12–13, 28, 227, 232, 235,
238, 246
persistent 11, 23, 25
planned takeover 117–118, 156–157, 160, 164,
182, 308
primary 5–6, 26, 34–36, 38, 41, 119–123, 234, 238,
247–249, 302–304, 306
primary key 25–26, 37–38
primary server 35–36, 42, 120–124, 247–249, 253,
302
primary server outage 44, 46–48
propagated changes 96, 98, 250
publishing queue maps 24–25

Q

Q Apply 10–12, 17–18, 43, 46–47, 50, 53, 144,
155–158, 226, 229, 231–232
Q Apply control tables 24
Q Apply latency 30, 32
Q Apply log 53, 55, 57–59, 157–158, 229
Q Capture 10–11, 14–15, 17, 43–44, 46–48,
143–144, 156, 159, 226, 231, 234, 249
Q Capture control tables 24
Q Capture latency 30–31, 60, 63, 79, 87, 113
Q Capture log 52–53, 58–59, 65, 159, 163, 172
Q Capture parameters 20
Q Capture transaction latency 30–32
Q latency 30–32
Q replication control tables 11, 115, 151–152
Q replication overview 9–10
Q replication topology xix, 9, 28

Q subscriptions 11, 14, 18, 20–21, 153–154
queue manager 12, 21, 23–25, 54, 66, 75, 88–89
quiesce 129

R

rationale for the bidirectional solution 33
rationale for the unidirectional solution 118–119
receive queue 23, 30
redundant data 4
re-establish pre-takeover environment 118
referential integrity 17–18, 25–27, 36, 276, 278–279, 291
Replication Center 14–15, 17–18, 22, 60, 62–64, 67, 151, 236, 239, 241–242
replication queue maps 11, 24, 153
restart information 23, 172
restart queue 24
resume primary workload 114
Resume RO workload 64, 80, 102, 109, 114
Resume RO workload on secondary 114
resync 46, 76, 84, 98, 102, 108
resync database 112

S

sample exceptions 225, 247
schema 15, 17–18, 20, 25, 226, 228, 240
search condition 20
secondary 5, 35–36, 42–43, 119, 124, 127, 131, 144, 247–249, 253
secondary server 35–36, 41–42, 124, 128, 135, 144, 146, 234, 247–249, 253
send queue 23
service recovery time 2
site failure 304
spill queue 24
SQL error exceptions 48, 71, 76, 80, 85, 250
SQL errors considerations 229, 236
standby 5, 11, 70, 121–122, 125, 128, 133, 302–304
standby server 70, 121–122, 125, 128, 134, 302
Start Q Apply 57, 59, 78, 86, 102, 157, 186
Start Q Apply on secondary 78, 104
Start Q Capture 58–59, 67, 78, 80, 159, 171, 186, 193, 214
Start Q Capture on primary 104
Start Q Capture on secondary 80, 86, 114
Stop Q Apply 75
Stop Q Capture 52, 75, 162, 187, 206

stop workload on secondary 64, 79, 87, 105, 112
switchback xix, 33–36, 41, 43–44, 46, 247, 249–250, 276–277, 308
switchback operations 45, 48, 277, 281
switchback procedure 45, 56, 76, 84
synchronization 13, 19, 122, 127, 132, 249, 302–303
system availability 2–3, 5
system failover 5

T

TAKEOVER 122, 139, 164–165, 188
takeover 5, 28, 117–118, 142, 144, 156, 305–306, 308
toggle 28
topologies 28, 35
transaction messages 24, 26, 31
transaction thread 31
triggering event 43

U

uncontrolled failover (UF) and switchback 70
uncontrolled failover and switchback 33–34
unidirectional replication xix, 12, 19, 117–120, 143, 145, 156, 185
unique constraint 25–26
unplanned takeover 117–118, 185–186, 195, 198, 308
unpropagated changes 43, 45–48, 187, 277, 281–283, 285

V

verify all workload changes applied to target tables 182, 195, 218
Verify Q replication up and running 80, 102, 109, 113, 178, 195, 218
Verify RECVQ 66, 74, 79, 88
Verify XMITQ 54, 65, 79, 87, 105
viewing exceptions in the IBMQREP_EXCEPTIONS table 229

W

warm starting 46–47, 76, 102
WebSphere MQ 10–12, 19–20, 42, 44, 54, 56, 66, 144, 148, 150, 280
WebSphere MQ client 25, 148, 150
WebSphere MQ objects 23–25, 114, 148

WebSphere MQ queues 12, 19, 23, 25, 44

X

XML format 21, 226

XML publication 20–22, 31, 228



WebSphere Replication Server Using Q Replication: High Availability Scenarios for AIX

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Redbooks

WebSphere Replication Server Using Q Replication

High Availability Scenarios for the AIX Platform

WebSphere Replication Server Q replication overview

Bidirectional Q replication failover/switchback scenarios

HADR and Q replication coexistence scenario

This IBM Redbook provides detailed instructions and scripts on managing failover and switchback in a bidirectional Q replication environment for the AIX platform. A typical business scenario is used to showcase the bidirectional failover/switchback implementation. The redbook also includes a HADR 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 redbook is organized into the following topics:

- ▶ High availability concepts
- ▶ WebSphere Replication Server Q replication overview
- ▶ Failover and switchback scenarios
- ▶ HADR and Q replication coexistence scenario
- ▶ Summary of code and scripts used in the scenarios
- ▶ Exception processing in a bidirectional Q replication environment
- ▶ Overview of HADR and DB2 Client Reroute

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks